AD-A092 010 GTE PRODUCTS CORP NEEDHAM HEIGHTS MA COMMUNICATION S--ETC F/G 9/4
SPEECH ALGORITHM OPTIMIZATION AT 16 KBPS.(U)
SEP 80 R S CHEUNG, S Y KWON, A J GOLDBERG DCA100-79-C-0038

UNCLASSIFIED NL

1 OF 3
AD A
092010

LEVEL II

AD A092010

FINAL REPORT

SPEECH ALGORITHM OPTIMIZATION AT 16 KBPS

DCA 100-79-C-0038

DTIC ELECTE NOV 7 1980 S C

SUBMITTED TO

DEFENSE COMMUNICATIONS AGENCY

SEPTEMBER 30, 1980

Sylvania Systems Group
Communication Systems Division
GTE Products Corporation
77 A Street
Needham Heights, Mass. 02194 U.S.A.
Area Code 617 449-2000
TELEX: 92-2497

GTE Systems

80 11 07 049

Unclassified

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

REPORT DOCUMENTATION PAGE

READ INSTRUCTIONS BEFORE COMPLETING FORM

1. REPORT NUMBER	2. GOVT ACCESSION NO. AD-A092010
3. RECIPIENT'S CATALOG NUMBER	
4. TITLE (and Subtitle) FINAL REPORT - SPEECH ALGORITHM OPTIMIZATION AT 16 KBPS. (6)	5. TYPE OF REPORT & PERIOD COVERED (9) Final Report, July 79 — Sep[redacted] 80,
	6. PERFORMING ORG. REPORT NUMBER
7. AUTHOR(s) (10) R. S. Cheung, S. Y. Kwon[redacted]d A. J. Goldberg	8. CONTRACT OR GRANT NUMBER(s) (15) DCA 100-79-C-0038
9. PERFORMING ORGANIZATION NAME AND ADDRESS GTE Products ~~Inc.~~ 77 "A" Street Needham Heights, MA 02194	10. PROGRAM ELEMENT, PROJECT, TASK AREA & WORK UNIT NUMBERS 12259
11. CONTROLLING OFFICE NAME AND ADDRESS Defense Communications Agency Contract Management Division, Code 260 Washington, D.C. 20305	12. REPORT DATE (11) 30 Sep[redacted]980
	13. NUMBER OF PAGES 249 pages
14. MONITORING AGENCY NAME & ADDRESS(if different from Controlling Office)	15. SECURITY CLASS. (of this report) Unclassified
	15a. DECLASSIFICATION/DOWNGRADING SCHEDULE

16. DISTRIBUTION STATEMENT (of this Report)

Distribution of this document is unlimited. It may be released to the Clearinghouse, Department of Commerce, for sale to the general public.

17. DISTRIBUTION STATEMENT (of the abstract entered in Block 20, if different from Report)

DTIC ELECTE NOV 7 1980 S D C

18. SUPPLEMENTARY NOTES

19. KEY WORDS (Continue on reverse side if necessary and identify by block number)

SBAPC, CVSD, quadrature mirror filters, noise shaping, adaptive bit allocation, split-band voice coding, adaptive predictive coding, noise suppression.

20. ABSTRACT (Continue on reverse side if necessary and identify by block number)

This report describes the design of a 16 Kb/s Split-Band Adaptive Predictive Coder (SBAPC) whose high quality processed speech compares favorably with that of the Continuously Variable Slope Deltamodulator (CVSD) in both back-to-back and simulated tactical situations. Operations of this algorithm include the partitioning of the input frequency band evenly into two subbands and the encoding of the subband waveform using Adaptive Predictive Coding (APC). The results of our investigation on (Cont'd)

DD FORM 1473 1 JAN 73 EDITION OF 1 NOV 65 IS OBSOLETE

Unclassified

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

Unclassified

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

20. ABSTRACT (Cont'd)

quadrature mirror filters, tradeoffs between various APC parameters, noise shaping techniques, and adaptive bit allocations are presented. This report also gives a detailed discussion on the utilization of noise suppression techniques in reducing stationary background noise and forward error-correcting codes in maintaining the SBAPC performance at 10^{-2} channel error rate.

10 to the minus 2

Unclassified

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

TABLE OF CONTENTS

Section		Page
	LIST OF ILLUSTRATIONS	iii
	LIST OF TABLES	v
I	SUMMARY	
	1.1 Summary of the Program	1-1
II	ALGORITHM DEVELOPMENT	
	2.1 Introduction	2-1
	2.2 Design of Quadrature Mirror Filters	2-4
	2.2.1 The Windowing Technique	2-6
	2.2.2 The Optimization Technique	2-13
	2.3 Adaptive Predictive Coding of Split Band Signals	2-17
	2.3.1 Adaptive Predictive Coder with One Loop	2-19
	2.3.2 Adaptive Predictive Coder with Two Loops	2-23
	2.3.3 Tradeoff Analysis of SBAPC Systems	2-32
	2.3.3.1 The Performance of the SBAPC System with One Loop	2-33
	2.3.3.2 The Performance of the SBAPC System with Two Loops	2-37
	2.4 The Shaping of Quantization Noise in SBAPC Systems	2-40
	2.4.1 Makhoul's Noise Shaping Technique	2-40
	2.4.2 Atal's Noise Shaping Technique	2-47
	2.4.3 Comparison of Atal's and Makhoul's Noise Shaping Methods	2-55
	2.5 Quantization of Residual Signals	2-59
	2.5.1 Adaptive Bit Allocations	2-59
	2.5.2 Quantization of Residual Signals	2-65
III	PERFORMANCE UNDER CHANNEL IMPAIRMENTS	
	3.1 The Effect of Background Noise on the SBAPC	3-1
	3.1.1 Reduction of the Background Noise	3-2
	3.1.2 McAuley's Noise Suppression Technique	3-3
	3.2 The Effect of Random Channel Errors on the SBAPC	3-17
	3.2.1 Application of BCH Codes	3-20
	3.2.2 Error Protection via the (127,106) BCH Code	3-22
	3.2.3 Error Protection via Five Blocks of (63,45) Codes	3-26
	3.3 Tandem Performance with 2.4 Kb/s LPC-10	3-26

TABLE OF CONTENTS (Cont'd)

Section		Page
IV	FORTRAN SIMULATIONS	
	4.1 FORTRAN Simulations of the SBAPC System	4-1
	4.2 The User's Guide	
	4.2.1 Task Building	
	4.2.2 Operating Procedures	
V	CONCLUSIONS AND RECOMMENDATIONS	
	5.1 Conclusions	5-1
	5.2 Recommendations	5-2
	REFERENCES	5-4
APPENDICES		
A	Theory of Quadrature Mirror Filters	A-1
B	Modified Robert's Noise Detection Algorithm	B-1
C	Primitive BCH Codes	C-1
D	Operations in Galois Field	D-1
E	Listings of FORTRAN Programs	E-1
F	The 16 Kb/s Adaptive Transform Coder	F-1
G	Interrelationships Between DCT and DFT Algorithms	G-1

LIST OF ILLUSTRATIONS

Figure		Page
2.1.1	Block Diagram of the Split-Band APC Technique	2-2
2.2.1	Magnitude Response of a 60-Tap QMF Designed Using the Hanning Window Method	2-9
2.2.2	Magnitude Response of an Unquantized SBAPC System with the 60-Tap QMF	2-10
2.2.3	Magnitude Response of a 32-Tap QMF Designed Using the Hanning Window Method	2-11
2.2.4	Magnitude Response of the Unquantized SBAPC System with the 32-Tap QMF	2-12
2.2.5	Magnitude Response of a 32-Tap QMF Designed Using the Optimization Procedure	2-15
2.2.6	Magnitude Response of an Unquantized SBAPC System with the 32-Tap QMF Designed Using the Optimization Technique	2-16
2.2.7	Comparisons of Unquantized SBAPC Systems with Two Different QMF's	2-18
2.3.1	Block Diagram of an APC System with One Loop	2-20
2.3.2	Block Diagram of an Adaptive Predictive Coder With Two Loops	2-24
2.3.3	Reconfiguration of the Analyzer of a Two-Loop APC to Minimize the Effect of Quantization	2-28
2.3.4	Block Diagram of the SBAPC System with One Loop	2-34
2.3.5	A Signal-to-Quantization Noise Ratio Plot of the 16 Kbps SBAPC System with One Loop	2-36
2.3.6	Block Diagram of a SBAPC System with Two Loops	2-38
2.4.1	Block Diagram of APC System	2-41
2.4.2	Block Diagram of the APC System with Noise Shaping	2-44
2.4.3	Block Diagram of the Generalized APC System with Atal's Noise Shaping Technique	2-48
2.4.4	Spectrum Plot of the SBAPC System with Time Domain SNR = 12.77 dB with ALPA = 0.0 to Both Band in Atal's Noise Shaping Technique	2-54
2.4.5	Spectrum Plot of the SBAPC System with Time Domain NSR = 21.23 dB with ALPA = 1. for Both Band Using Atal's Technique	2-56
2.5.1	The Performance Curves of a SBAPC System with or Without Quantization of Error Signals	2-60
2.5.2	The Performance Plots of 16 Kbps SBAPC Systems with Fixed and Adaptive Bit Allocations	2-64
2.5.3	Distribution of the Low Band Residual Signal	2-68

LIST OF ILLUSTRATIONS (Cont'd)

Figure		Page
2.5.4	Distribution of the High Band Residual Signal	2-69
3.1.1	Transfer Characteristics of the Noise Suppression Device	3-8
3.1.2	The Characteristic Function of the Maximum Likelihood Noise Suppression	3-12
3.1.3	Block Diagram of McAuley's Noise Suppression Algorithm	3-14
3.2.1	The 16 Kbps Split-Band APC System	3-18
3.2.2	The Effect of Channel Error Rates on 3 SBAPC Systems	3-19
4.1.1	Flow Diagram of the 16 Kbps SBAPC FORTRAN Program	4-2
4.1.2	Flowchart of Noise Suppression Routine	4-3
	Flowchart of Noise Suppression Routine (Cont'd)	4-4
4.1.3	Flowchart of QMF with Down-Sampling	4-6
4.1.4	Flowchart of Low-Band Predictor Coefs Computation	4-7
4.1.5	Flowchart of High-Band Predictor Coefs Computation	4-8
4.1.6	Flowchart of Adaptive Bit Allocation	4-9
4.1.7	Flowchart of Low-Band APC Analyzer with Noise Sharing	4-10
4.1.8	KT-th Block Encoding Routine for (63,45) BCH Code in 16 Kbps SBAPC System	4-11
4.1.9	Flowchart of Low-Band APC Synthesizer	4-13
4.1.10	Flowchart of QMF with Up-Sampling	4-14
4.2.1	Print-Outs of SBAPC Program (ξ=0)	4-17
4.2.2	Print-Outs of SBAPC Program ($\xi=\theta$)	4-18
A.1	Band-Splitting and Reconstruction Using QMF	A-2
B.1	Modified Robert's Noise Detection Algorithm	B-2
C.1	Computation of σ_1, σ_2, σ_3	C-8
C.2	Chien's Search Decoding Procedure	C-10
F.1	Discrete Cosine Transform Operation	F-1
F.2	Adaptive Transform Coder	F-4
F.3	Graphical Description of Vocoder Strategy for ATC	F-8
	Graphical Description of Vocoder Strategy for ATC (cont)	F-9

LIST OF TABLES

Table		Page
1-1	Optimized SBAPC System Specification	1-3
2-1	Tabulation of the 32-Tap QMF Designed Using the Optimization Method	2-14
2-2	Signal-to-Quantization Noise Ratios of the SBAPC System with One Loop at 16 Kbps	2-35
2-3	Signal-to-Quantization Noise Ratio of a SBAPC System with Two Loops at 16 Kbps	2-39
2-4	Signal to Noise Ratio of SBAPC System with Makhoul's Noise Shaping at 8 Kbps	2-46
2-5	The Signal-to-Quantization Noise Ratio in dB of the SBAPC System with Atal's Noise Shaping at 8 Kbps	2-53
2-6	Comparison of SBAPC Systems with Two Different Noise Shaping Techniques	2-58
2-7	Signal-to-Quantization Noise Ratio of the Two Loop SBAPC System With and Without Adaptive Bit Allocations at 16 Kbps	2-62
C-1	Generator Polynomials for Selected Primitive BCH Codes	C-3
F-1	16 Kbps ATC System Specification	F-10

SECTION I

SUMMARY

1.1 Summary of the Program

Under the fifteen-month Speech Algorithm Optimization at 16 Kb/s Contract (DCA 100-79-C-0038), GTE developed a FORTRAN simulation of the Split-Band Adaptive Predictive Coder (SBAPC) which yielded high quality speech outputs at 16 Kb/s. This software was designed to run on a PDP-11 computer with the FORTRAN IV-PLUS compiler.

The study has resulted in a number of significant accomplishments for developing speech processing algorithm at 16 Kb/s. Among them, the most important ones are:

1) the development of the high quality 16 Kb/s Split-Band APC algorithm which includes:

   i) the design of Quadrature Mirror Filters (QMF)

   ii) the tradeoffs between APC parameters

   iii) comparisons of Atal and Makhoul's noise shaping techniques

   iv) adaptive allocation of bits between the subbands

2) the incorporation of McAuley's noise suppression scheme in the SBAPC which yields intelligible speech even in a noisy background of -6 dB signal-to-noise ratio

3) the design of forward error correction codes that maintain the performance of the Split-Band APC algorithm in the presence of 10^{-2} channel error rate.

This Speech Algorithm Optimization at 16 Kb/s Contract was partly motivated by the fact that high quality communications at 16 Kb/s represent one of the government's long-term goals. Moreover, the Continuously Variable Slope Deltamodulation (CVSD) scheme presently employed by the 16 Kb/s Tenley terminals does not produce quality outputs that satisfy all communication needs. The performance of these Tenley terminals, when in tandem with the STU-2's, is further compromised by the interactions of CVSD's granular and slope-overloading characteristics with the buzzy quality of the 2.4 Kb/s Linear Predictive Coder (LPC). Consequently, an improved speech encoding scheme is needed for future 16 Kb/s terminals. Studies in the past had resulted in algorithms that yielded much improved speech quality over CVSD at 16 Kb/s. An existence proof is given by the Adaptive Predictive Coder with Adaptive Quantization (APCQ) developed for DCA by GTE Sylvania under Contract No. DCA-100-76-C-0002. However, though these algorithms may yield outputs that compare favorably with CVSD in the back-to-back mode, yet the latter has been proven time after time to be one of the most robust algorithms under extremely adverse conditions. As a result, the 16 Kb/s CVSD technique still finds utility in the high noise environment of flight decks and the high error surroundings of mobile radios.

In this study, a 16 Kb/s Split-Band Adaptive Predictive Coder (SBAPC) has been investigated and our results indicate that the technique can indeed produce much improved speech quality over that of CVSD. Specifications of the 16 Kb/s SBAPC system is shown in Table 1-1. Basically, the algorithm calls for the splitting of the input frequency band using 32-tap Quadrature Mirror Filters (QMF) followed by the adaptive predictive coding (APC) of the subband waveforms. The windowing and the optimization techniques have been applied to the design of QMF's. Both procedures have

PARAMETER	SPECIFICATION
Input Bandwidth	0-3200 Hz
Sampling Rate	6400 Hz
Frame Rate	44.444/sec.
Number of Samples/Frame	144
Number of Samples Overlapped/Frame	18
Bits/Frame	360
Low Band Residual Energy	5
High Band Residual Energy	5
Low Band Pitch	6
Low Band Pitch Gain	4
Low Band PARCOR 1	5
2	5
3	3
4	3
High Band PARCOR 1	4
2	4
3	3
4	3
Residual Error Signals	216
Parity Bits (Error Correction)	90
SYNC	4
Number of Error Control Blocks/Frame	5
Error Control Technique	(63,45) BCH

TABLE 1-1: OPTIMIZED SBAPC SYSTEM SPECIFICATION

yielded QMF's that perform the bandsplitting and reconstruction with relatively little distortions. Frequency response and S/Q plots of the unquantized SBAPC system employing these filters have indicated that the QMF designed using the optimization technique is a slightly better one. For the low-band where pitch and first formant are present, our tradeoff analysis has shown that the APC with a first order pitch loop and a fourth order prediction loop is capable of preserving these perceptually important parameters. For the high-band where most unvoiced sounds, e.g., fricatives occur, a simple fourth-order prediction loop can be employed without hurting the overall quality. To account for the fluctuations of the distribution of energies between the high and low bands from frame to frame, a bit allocation scheme has been devised to dynamically alter the quantizer bit assignment. With the average assignment of 1.5 bits per sample, an improvement of 1 dB S/Q has been realized with the adaptive method. To further improve the speech quality, noise shaping algorithms have been incorporated in the SBAPC. In particular, the performance of Atal's and Makhoul's noise shaping techniques have been compared. Though the S/Q yielded by the two methods are roughly the same, Makhoul's second order all-zero shaping filter has resulted in higher quality speech.

Cognizant of the fact that CVSD yields intelligible speech under tactical situations, the performance of the SBAPC system has also been studied in the presence of background noise, channel errors and in tandem with LPC. In a low-noise office environment, the SBAPC yields high quality processed speech similar to that of the back-to-back mode. For high-noise surroundings (e.g., S/N = -6 dB), the algorithm still results in highly intelligible speech, but the noisy background makes it very annoying to listen to. To improve this, a noise reduction technique has been designed which works as a pre-processor to the SBAPC system. Depending on the signal-to-noise ratio of the additive noise, a suppression factor can be pre-determined to reduce its level. The integrated SBAPC algorithm with the pre-processor has yielded highly intelligible speech without much of the annoying background noise at S/N = -6 dB. Also, the SBAPC system has been found to be extremely sensitive to channel errors. Degradations in processed speech become noticeable at the bit error rate (BER) of 5×10^{-4} and the system yields unintelligible speech at 10^{-2} BER. As expected, the side information which contains the quantized pitch, PARCOR coefficients etc. is more sensitive to channel errors than the low and high band residual signals. In fact, one or two errors occurring on the side information can result in a frame of erroneous data. Furthermore, the pitch loop in the SBAPC algorithm compounds the effect by propagating these errors over several frames. In this study, forward error correcting codes have been found to be a viable solution to maintain the system performance over a high error rate channel. Particularly, five blocks of (63,45) BCH codes have been incorporated in the 16 Kb/s SBAPC scheme, and the overall system is robust to channel error rates as high as 10^{-2}. When in connection with LPC, the SBAPC algorithm has yielded more intelligible speech than the CVSD/LPC tandem.

Informal listening tests indicate that the SBAPC system yields much higher speech quality than that of CVSD in a back-to-back mode. Also, when compared to the 16 Kb/s adaptive transform coder (ATC) (a discussion is included in Appendix F), the SBAPC processed speech is slightly low-passed, but its smooth quality is much preferred over that of ATC with the noticeable "dish-washing" background noise. Furthermore, the SBAPC system has performed well in simulated tactical situations. Based on the results obtained in this study, further work should be performed to refine the algorithm and implement it in real-time.

SECTION II

ALGORITHM DEVELOPMENT

2.1 Introduction

The Split-Band APC technique is basically a combination of two speech coding methods, namely, the Adaptive Predictive Coding (APC) and Subband coding (SBC). The APC method is well known for its efficiency in processing speech waveforms in the time domain whereas the SBC is one of the simplest techniques in encoding speech in the frequency domain. By combining these two techniques together, the SBAPC algorithm yields high quality processed outputs without much additional complexity. In practice, the SBAPC technique calls for the partitioning of the input frequency band into two even subbands using Finite-Impulse-Response (FIR) filters followed by the application of different quantizations to the two subbands. Since the pitch and the first formant are located at the low frequency band, more detailed description of this band's waveform preserves these perceptually important parameters which can result in higher speech quality. On the other hand, the upper frequency band at which the unvoiced sounds, such as fricatives, are situated can be encoded less precisely without hurting the overall processed speech quality. Consequently, the SBAPC algorithm represents one of the most efficient methods in speech coding.

The block diagram of a Split-Band APC system is shown in Figure 2.1.1. Bandpass filters are utilized to split the input speech frequency band into two. Then each subband signal, after resampling at its Nyquist rate, is encoded using APC. At the receiver, the digital data is decoded using APC. After up-sampling, the original subband signals are created and the difference between them forms a replica of the original

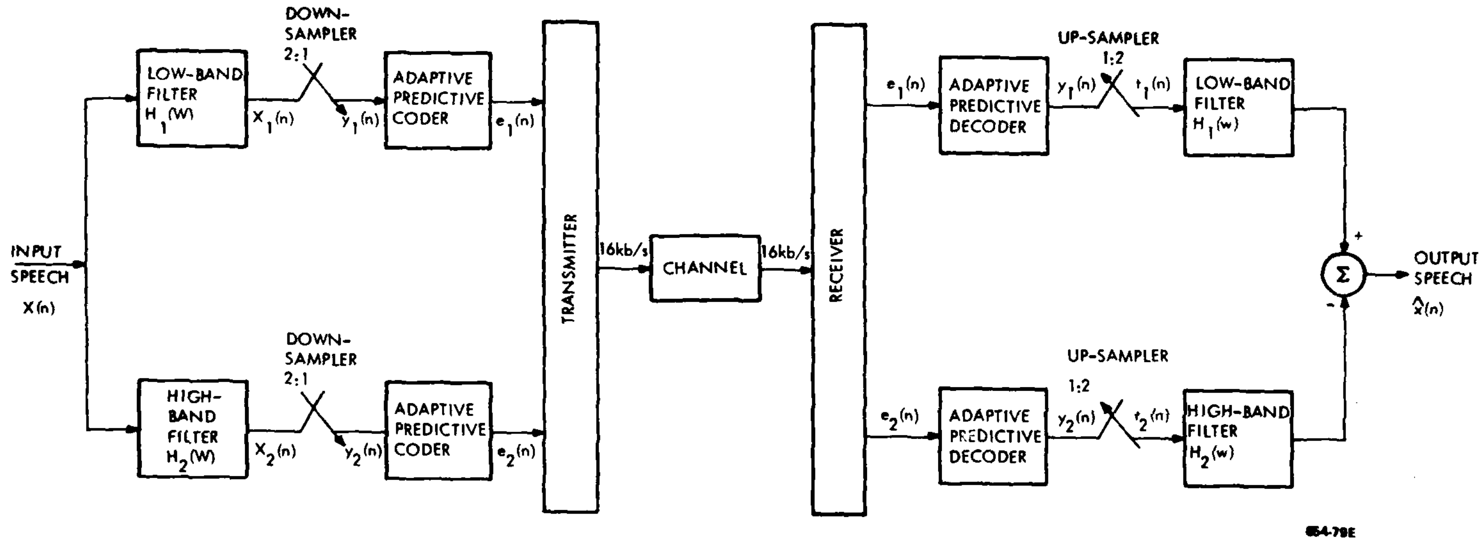


FIGURE 2.1.1: BLOCK DIAGRAM OF THE SPLIT-BAND APC TECHNIQUE

signal. The two subband signals, derived from splitting the original signal band, appear essentially as waveforms with non-flat spectral densities and contain a considerate amount of sample-to-sample correlations within each individual band. For these correlated signals, the APC technique, which attempts to minimize the rms error of the coded signal, is an efficient method of encoding the speech waveform into digital form. In fact, operations of the APC algorithm include the prediction of the past history of the waveform and the coding of the residual error signal formed by subtracting the estimate from the input speech. In this algorithm, efficient encoding is achieved because quantization is only applied to the residual error signal and prediction parameters which have significantly less dynamic range and sample-to-sample correlation as compared to the original signal. Moreover, the distortion from the quantization of the residual signal is the major source of speech degradation. In general, the power of the quantization noise is proportional to the power of the residual error signal. Thus, accurate prediction is essential to the minimization of this quantization error. Although small quantization error does not always mean small distortion perceptually, it generally leads to the production of a high quality synthesized speech. The following sections will discuss the design of a special class of split-band filters known as Quadrature Mirror Filters, the encoding of both low and high bands using APC coders, noise shaping algorithms, and the quantization of the residual waveforms.

2.2 Design of Quadrature Mirror Filters

A straightforward way of achieving band splitting is to perform band-pass filtering with the translation of the resulting signal spectrum to DC. This spectral translation can be accomplished in a variety of ways and with varying cost factors concerning efficiency, spectral distortion, and ease of implementation. The most common method is via integer-band sampling where the original signal, after filtering into several frequency bands of bandwidths f_i using FIR filters, is resampled at $2f_i$. To minimize distortions introduced by the band-splitting process alone, filter characteristics such as flat passband response, high stop-band attenuation, and short transition region are extremely desirable. To satisfy the above requirements, the FIR filters required is often of large order, thus increasing the amount of processing tremendously.

Recently, Quadrature Mirror Filters (QMF) have been successfully applied to the split-band or subband coding of speech signals [1, 2]. It has been shown that these QMF's can assure the perfect band-splitting and reconstruction of the input signals regardless of the filter length. For the sake of completeness, a discussion of the theory of QMF is included in Appendix A. It illustrates the fact that if the half-band filters satisfy constraints as defined in Eqs. (A-17 to A-19), no spectral distortion is introduced in the band-splitting and reconstruction processes. However, a filter that fulfills exactly all the QMF constraints is useless for a split-band coding scheme. To illustrate this, rewriting Eq. (A-13) in the Appendix A yields:

$$\hat{X}(Z) = 1/2\left[H_1^2(Z) - H_2^2(Z)\right]X(Z) \qquad (2\text{-}1)$$

where $\hat{X}(Z)$, $X(Z)$, $H_1(Z)$, $H_2(Z)$ are the Z-transforms of the output, the input, the lowband filter, the highband filter, respectively. Defining the transfer function of the overall QMF structure as H_{QMF}, its Z-transform is given by:

$$H_{QMF}(Z) = 1/2\left[H_1^2(Z) - H_2^2(Z)\right] \tag{2-2}$$

Equivalently, its impulse response is shown as:

$$h_{QMF}(n) = 1/2\left(h_1(n) \otimes h_1(n) - h_2(n) \otimes h_2(n)\right) \tag{2-3}$$

where $\otimes$ denotes the convolution operation. If $h_1(n)$ is represented by its even and odd parts,

$$h_1(n) = h_{1_e}(n) + h_{1_o}(n) \tag{2-4}$$

$h_2(n)$ defined as

$$h_2(n) = (-1)^n\ h_1(n) \qquad n = 0, 1, \ldots N-1 \tag{2-5}$$

can be written as:

$$h_2(n) = h_{1_e}(n) - h_{1_o}(n) \tag{2-6}$$

Substituting Eqs. (2-4) and (2-6) into Eq. (2-3), h_{QMF} becomes:

$$h_{QMF}(n) = 2\ h_{1_e}(n) \otimes h_{1_o}(n) \tag{2-7}$$

As dictated by the QMF constraints, the perfect bandsplitting and reconstruction requires $H_{QMF}(Z) = 1$ which means:

$$h_{QMF}(n) = \begin{cases} 1 & ;\ n=0 \\ 0 & ;\ n \neq 0 \end{cases} \tag{2-8}$$

From Eq. (2-7), this is only valid if the filter $h_1(n)$ has two non-zero and identical tap values. Hence, a true QMF can indeed be designed, but its two tap values generally do not yield sharp cutoff characteristics. Moreover, split-band coding techniques really do not offer any advantages over the full-band schemes unless the subbands have distinctly different frequency characteristics. This implies that the split-band filter should be a higher order one which has a faster roll-off in addition to meeting most of the requirements as stated in Eq. (A-17) to (A-19) of Appendix A.

In general, the half-band filters $h_1(n)$, are difficult to design when using conventional computer-aided design algorithms in order to satisfy all the QMF constraints. Of all well-known classes of digital filter design techniques, the McClellan-Parks algorithm is the most popular [3]. This algorithm, which uses the Remez exchange procedure, introduces several extremes in both pass and stopband regions. Although excellent stopband rejections can be achieved, the ripples cannot be guaranteed to combine in such a way that Eq. (A-19) is satisfied. In addition, since the 3-dB point cannot be designed easily to be at the half-band frequency, symmetry requirements are not easily attanied. To minimize the interaction between the passband and the stopband of the SBAPC system, design techniques that yield lowpass filters with a smooth passband response which fall off sharply after the 3-dB point and which have high stopband rejection have to be investigated.

2.2.1 The Windowing Technique

One simple approach to design lowpass filters that possess the above characteristics is to use the windowing technique. Depending on the type of windows utilized, short transition regions can be realized with a relatively low filter order. To illustrate the method, let's start with the frequency response of an ideal lowpass filter given as [3]:

$$H_d(e^{jw}) = \begin{cases} e^{-j\alpha} & ; |w| < w_c \\ 0 & ; \text{o.w.} \end{cases} \tag{2-9}$$

where α is the delay of the filter and w_c is the cutoff frequency in radians. Then the impulse response of the ideal filter is given by the inverse Fourier transform of $H_d(e^{jw})$ as:

$$h_d(n) = \begin{cases} \frac{1}{2\pi} \int_{-Wc}^{Wc} e^{jw(n-\alpha)} dw & ; n = \alpha \\ \sin \frac{Wc(n-\alpha)}{\pi(n-\alpha)} & ; n \neq \alpha \end{cases} \tag{2-10}$$

In order to design a finite-impulse-response filter with a zero phase delay, a window function $W(n)$ is applied to $h_d(n)$ as follows:

$$h(n) = h_d(n)\ W(n) \tag{2-11}$$

and α is selected as

$$\alpha = \frac{N-1}{2} \tag{2-12}$$

where N is the desired filter length.

There has been considerable research done on the design of digital filters with various window functions $W(n)$. For these windows, the tradeoffs between different filter parameters, such as the passband response, transition region, stopband rejection, etc., have been well documented [4]. Among them, the Hanning window which results in filters with a large stopband rejection and a short transition region with relatively low filter order can be applied to designing QMF.

In this case, the window function W(n) given as:

$$W(n) = \frac{1}{2}\left(1 - \cos\left(\frac{2\pi n}{N-1}\right)\right); \qquad 0 \leq n \leq N-1 \qquad (2\text{-}13)$$

is multiplied with $h_d(n)$ as depicted in Eq. (2-10). In order to satisfy the QMF constraints, N has to be even and w_c has to be determined experimentally to cutoff at the half-band frequency. Since the two end-points of the Hanning window are zeros, a window length of (N+2) points is employed to design a Nth order QMF.

Frequency responses of a 60-tap and a 32-tap QMF designed using the Hanning window are shown in Figures 2.2.1 and 2.2.3. Both filters exhibit flat passband responses of less than 0.07 dB ripple, large stopband rejections in excess of 50 dB and cutoff frequencies at the half-band point (1600 Hz). Though the 32-tap filter has twice the transition bandwidth (400 Hz) as compared to the 60-tap one, the utility of the shorter filter in the SBAPC algorithm represents a good compromise between complexity and performance. Employing these filters, the frequency responses of unquantized SBAPC systems are shown in Figures 2.2.2 and 2.2.4. Both graphs depict a uniform response of less than 0.4 dB ripple which further substantiate the fact that these Hanning filters satisfy most of the QMF constraints and they do not introduce much distortion in the band-splitting and reconstruction process.

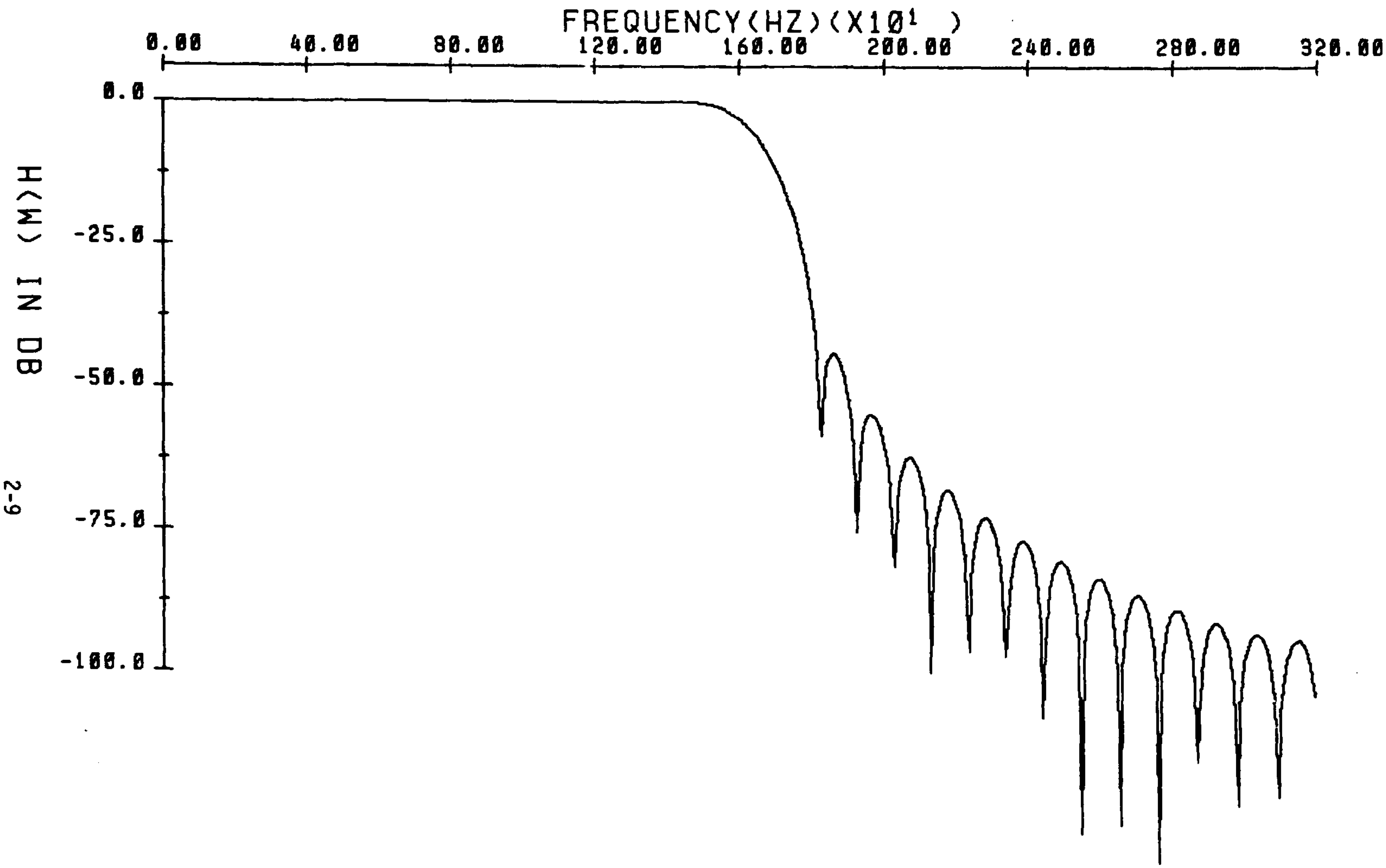


FIGURE 2.2.1: MAGNITUDE RESPONSE OF A 60-TAP QMF DESIGNED USING THE HANNING WINDOW METHOD

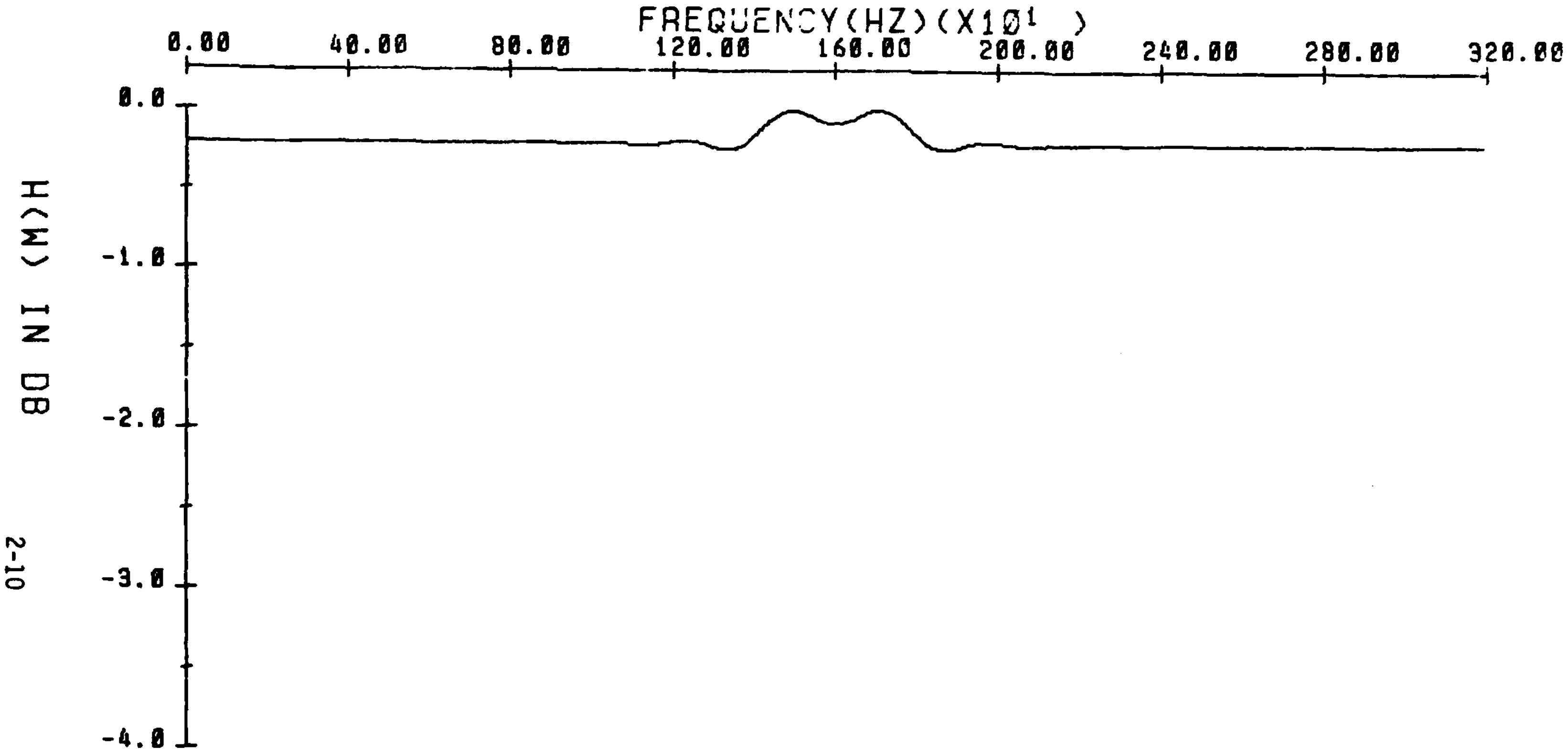


FIGURE 2.2.2: MAGNITUDE RESPONSE OF AN UNQUANTIZED SBAPC SYSTEM WITH THE 60-TAP QMF

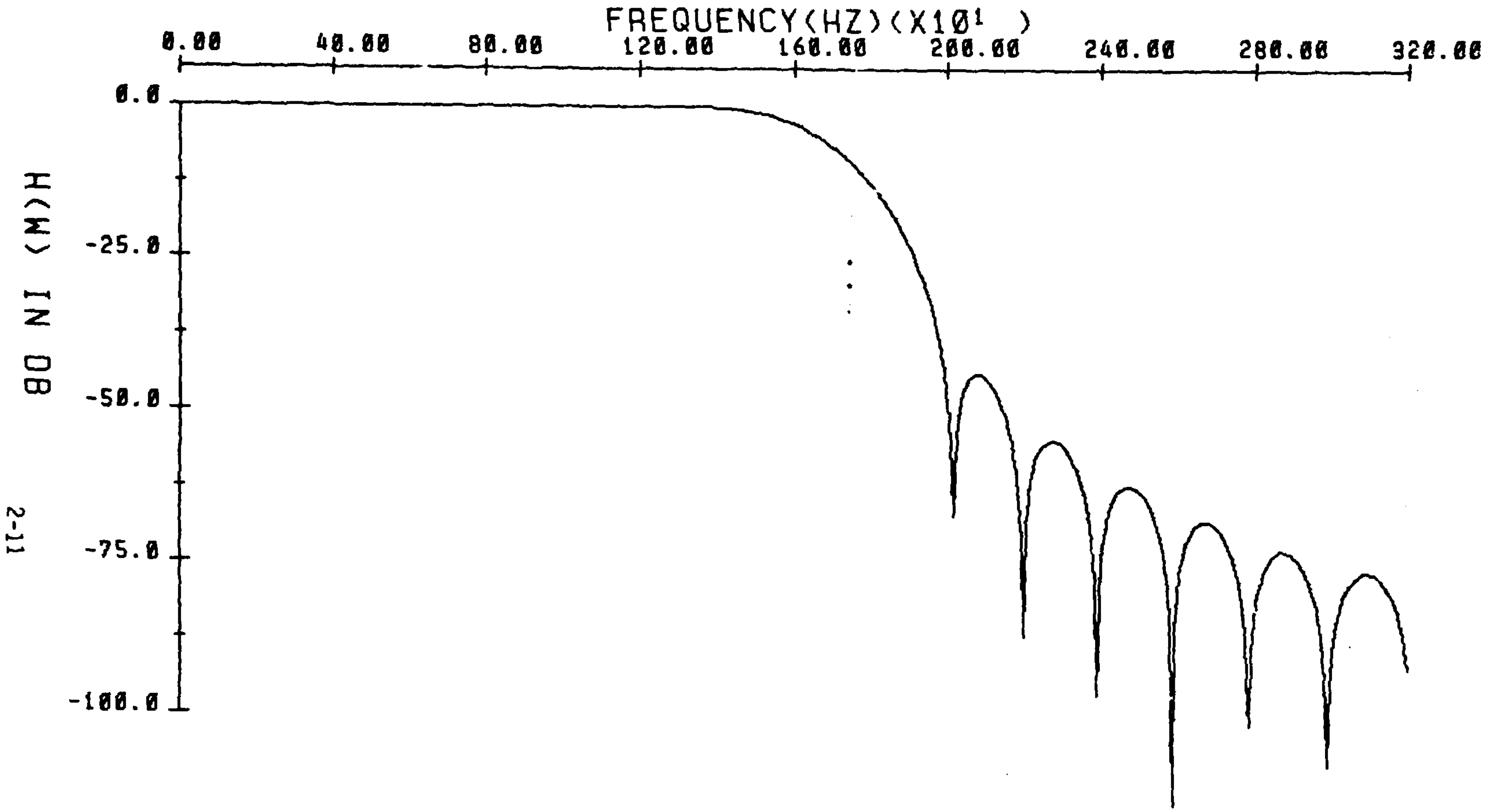


FIGURE 2.2.3: MAGNITUDE RESPONSE OF A 32-TAP QMF DESIGNED USING THE HANNING WINDOW METHOD

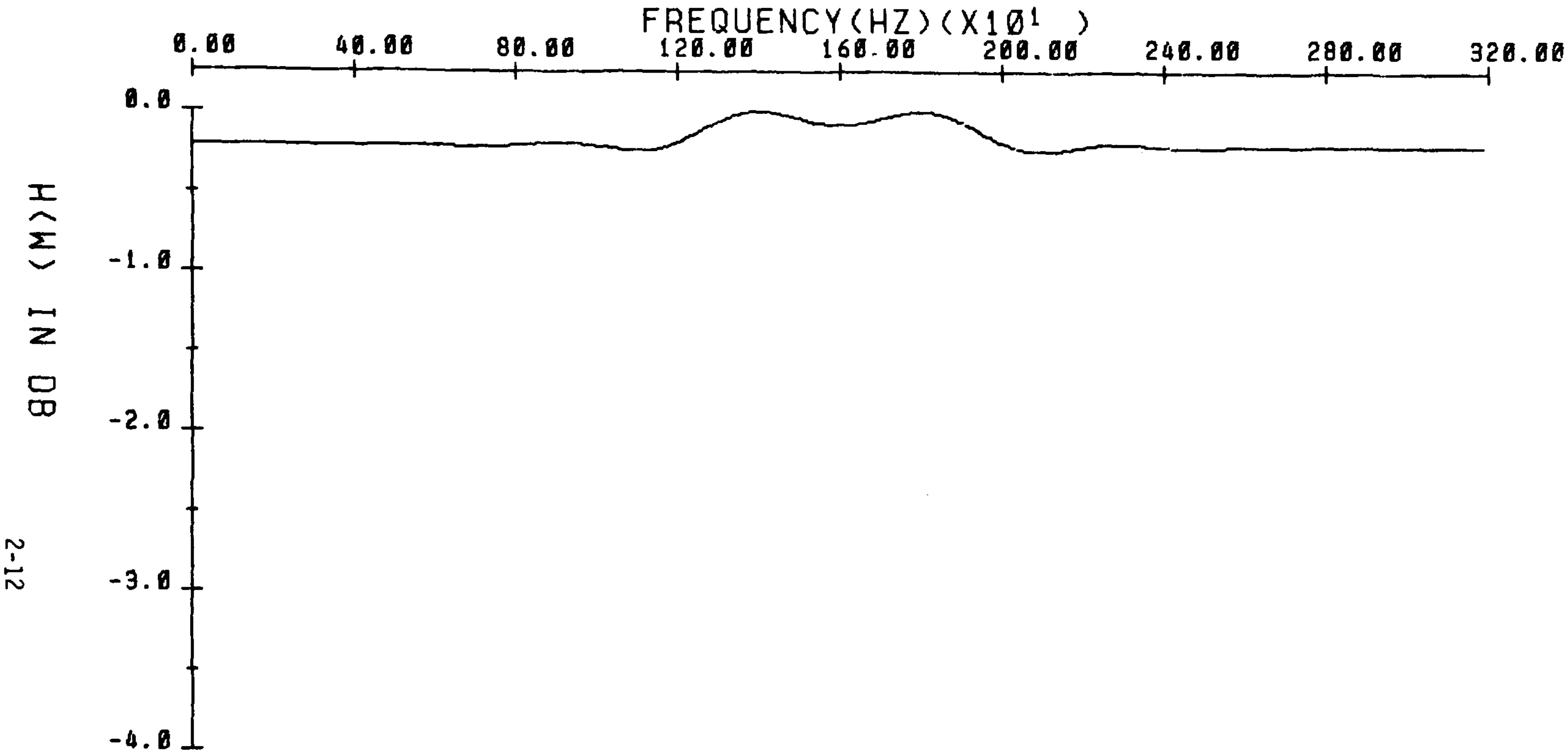


FIGURE 2.2.4: MAGNITUDE RESPONSE OF THE UNQUANTIZED SBAPC SYSTEM WITH THE 32-TAP QMF

2.2.2 The Optimization Technique

As discussed in Section 2.2.1, the Hanning window is a very simple technique to design filters that perform like QMF's. Though the magnitude response of the unquantized SBAPC system that utilizes these filters still exhibits a 0.4 dB "hump," it is nevertheless a good starting point in designing true QMF's. Recently, an optimization procedure has been proposed which results in QMF through the minimization of a performance index defined as [5]:

$$E = E_r + \alpha \, Es(f_{SB}) \tag{2-14}$$

where α, f_{SB} are the weighting, the frequency of the stopband; E_r is the ripple energy given as:

$$E_r = 2 \sum_{w=0}^{\pi/2} \left(H_1^2(w) + H_1^2(\pi - w) - 1 \right) \tag{2-15}$$

and E_s is the out-of-band energy given as:

$$E_s(f_{SB}) = \sum_{w=f_{SB}}^{\pi} H_1^2(w) \tag{2-16}$$

Utilizing the filter coefficients obtained through the Hanning window scheme as a starting point, an iterative search algorithm is formulated in locating the local minimum of E. QMF's designed using this method have been tabulated and an example of a 32-tap filter is included in Table 2-1. The magnitude response of the filter and the unquantized SBAPC system employing this filter are shown in Figure 2.2.5 and Figure 2.2.6, respectively. As compared to the 32-tap Hanning filter as depicted in Figures 2.2.3 and 2.2.4, the new filter is greatly improved in the sense that it exhibits a flatter unquantized SBAPC sys-

$h_1(0) = +0.69105790E-03 = h_1(31)$

$h_1(1) = -0.14037930E-02 = h_1(30)$

$h_1(2) = -0.12683030E-02 = h_1(29)$

$h_1(3) = +0.42341950E-02 = h_1(28)$

$h_1(4) = +0.14142460E-02 = h_1(27)$

$h_1(5) = -0.94583180E-02 = h_1(26)$

$h_1(6) = -0.13038590E-03 = h_1(25)$

$h_1(7) = +0.17981450E-01 = h_1(24)$

$h_1(8) = -0.41874830E-02 = h_1(23)$

$h_1(9) = -0.31238620E-01 = h_1(22)$

$h_1(10) = +0.14568440E-01 = h_1(21)$

$h_1(11) = +0.52947450E-01 = h_1(20)$

$h_1(12) = -0.39348780E-01 = h_1(19)$

$h_1(13) = -0.99802430E-01 = h_1(18)$

$h_1(14) = +0.12855790E+00 = h_1(17)$

$h_1(15) = +0.46640530E+00 = h_1(16)$

TABLE 2-1: TABULATION OF THE 32-TAP QMF DESIGNED USING THE OPTIMIZATION METHOD

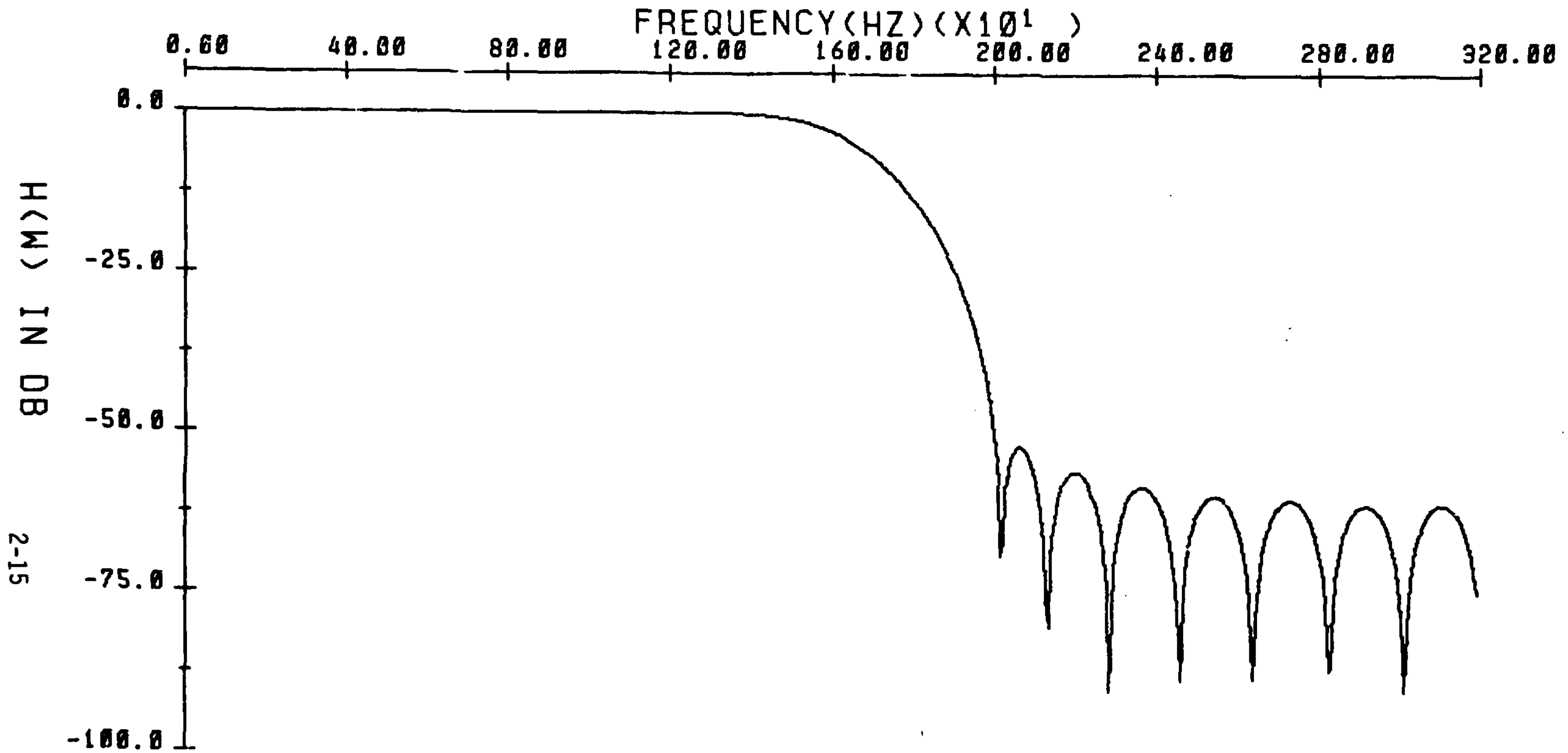


FIGURE 2.2.5: MAGNITUDE RESPONSE OF A 32-TAP QMF DESIGNED USING THE OPTIMIZATION PROCEDURE

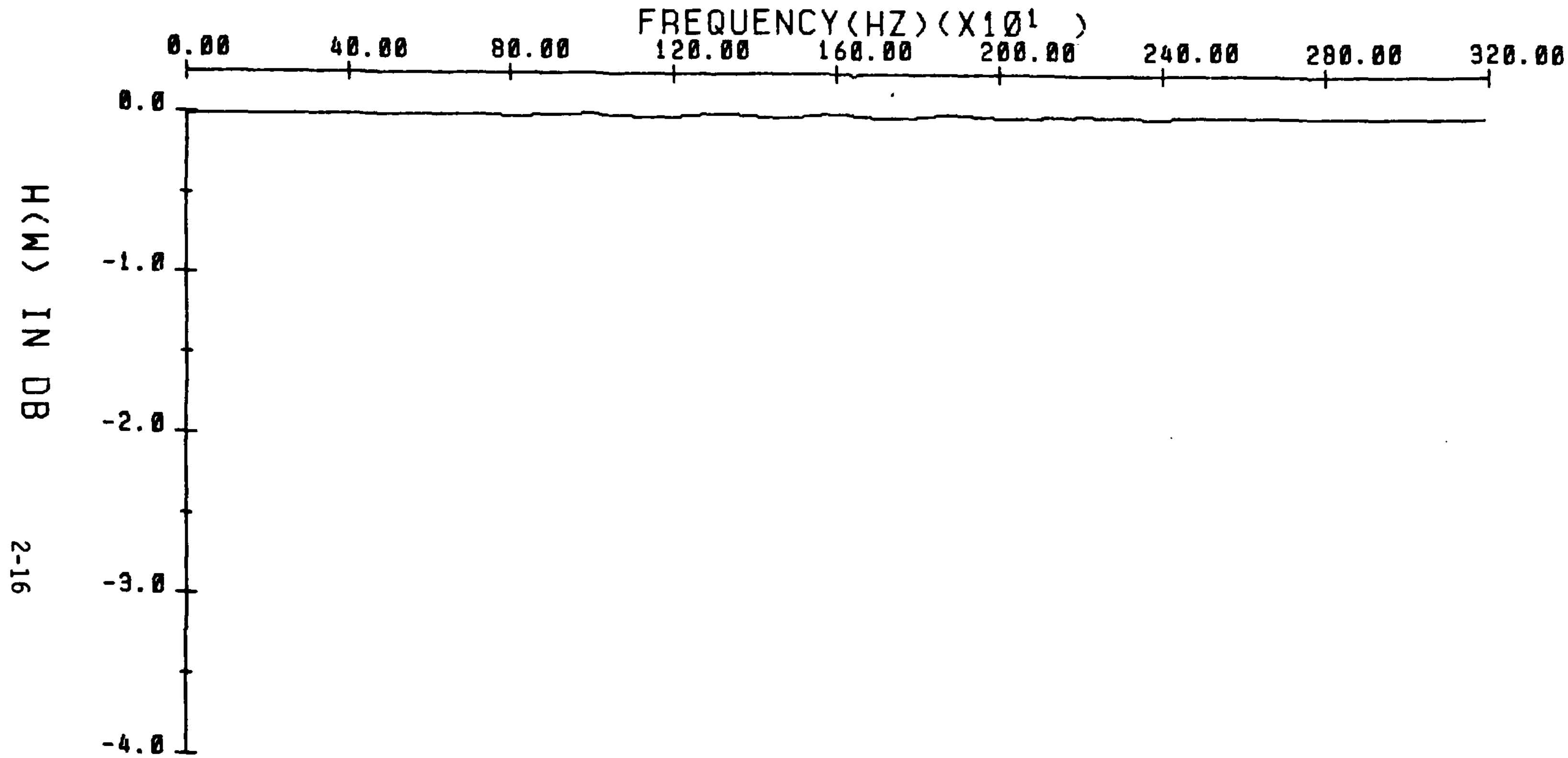


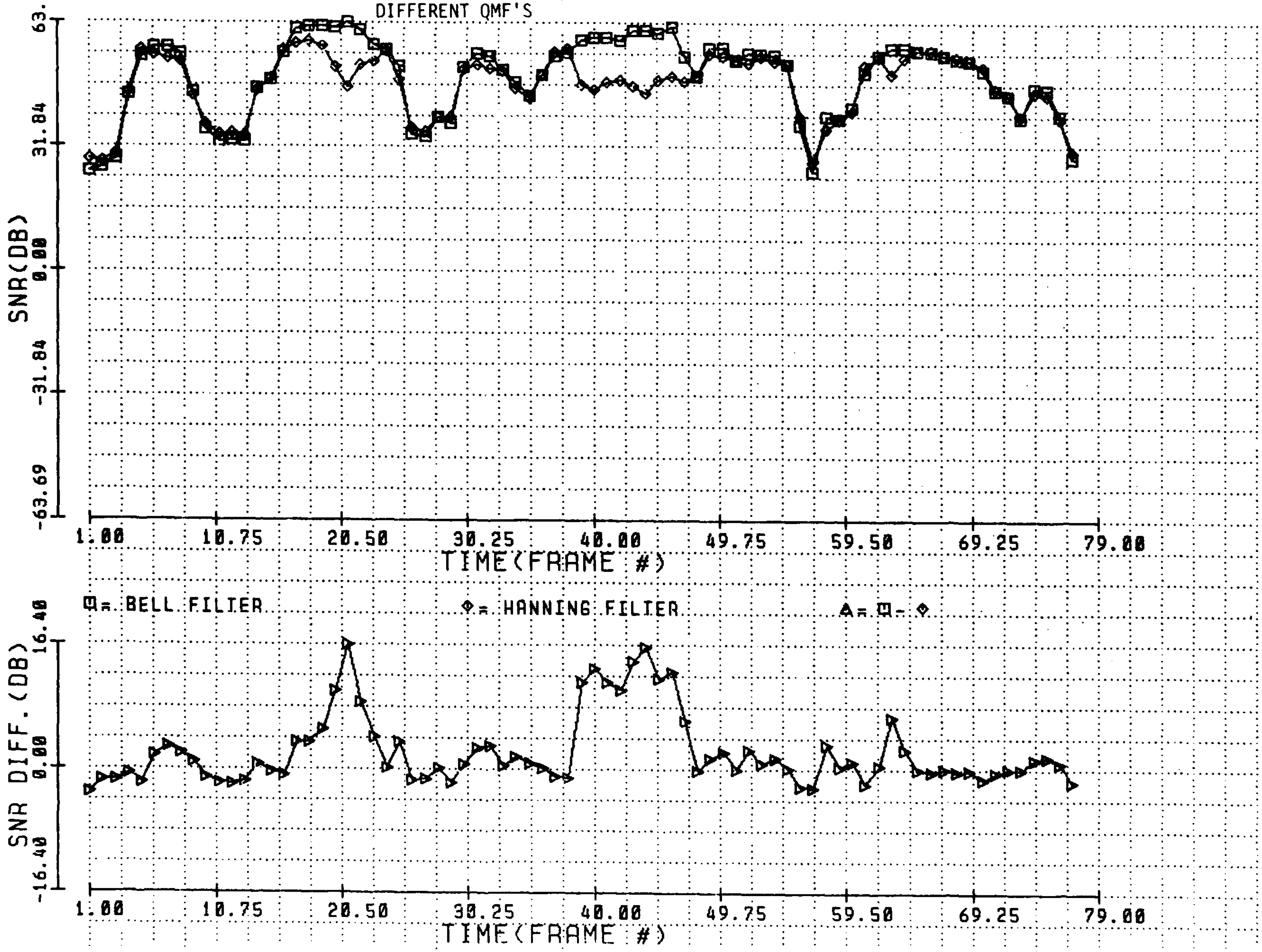
FIGURE 2.2.6: MAGNITUDE RESPONSE OF AN UNQUANTIZED SBAPC SYSTEM WITH THE 32-TAP QMF DESIGNED USING THE OPTIMIZATION TECHNIQUE

tem response without sacrificing much transition bandwidth and stopband rejection. Though informal listening tests conducted on the unquantized SBAPC processed sentences with the two 32-tap QMF's indicate no audible differences, S/N ratio plots of these systems, as depicted in Figure 2.2.7, show that the new filter consistently out-performs the Hanning filter especially during voiced regions. As a result, the 32-tap filter designed using the optimization procedure is employed in the subsequent studies.

2.3 Adaptive Predictive Coding of Split-Band Signals

APC algorithms have been extensively studied and reported on in the literature [6] - [8]. Most systems differ from one to another in the manner of parameter extractions and the design of quantizers with various level of complexity. In this section, the design of adaptive predictive coders, which minimizes the power of quantization error or maximizes the signal-to-quantization noise (S/Q), will be considered.

FIGURE 2.2.7: COMPARISONS OF UNQUANTIZED SBAPC SYSTEMS WITH TWO DIFFERENT QMF'S



2.3.1 Adaptive Predictive Coder with one Loop

The APC system with one loop is shown in Figure 2.3.1. In this scheme, the estimate of the present speech sample is assumed to be

$$\hat{s}_n = \sum_{i=1}^{P} a_i S_{n-i} + b_1 S_{n-M+1} + b_2 S_{n-M} + b_3 S_{n-M-1} \tag{2-17}$$

where M represent the number of speech samples in one pitch period and P is the predictor order in the prediction loop. Here, we consider a 3rd order pitch predictor, although a first order pitch predictor is common in conventional APC systems. The difference between the input speech sample s_n and the estimate $\hat{s}_n$ can be expressed as:

$$e_n = s_n - \hat{s}_n \tag{2-18}$$

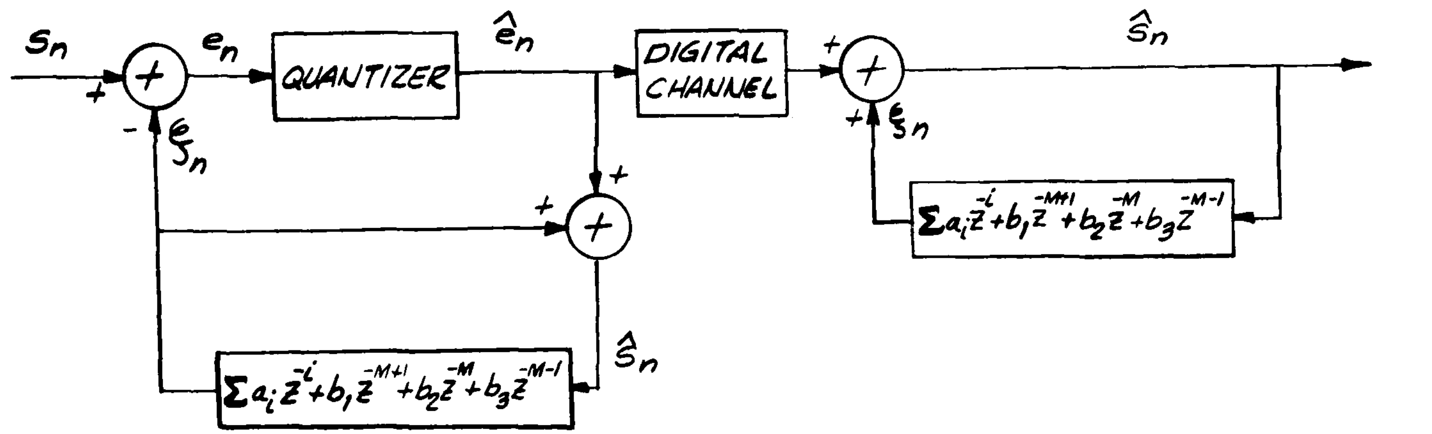
The total squared error may be shown as:

$$E = \sum_{n=1}^{L} e_n^2$$

$$= \sum_{n=1}^{L} \left[s_n - \left(\sum_{i=1}^{P} a_i S_{n-i} + b_1 S_{n-M+1} + b_2 S_{n-M} + b_3 S_{n-M-1} \right) \right]^2 \tag{2-19}$$

where L is the number of samples within a frame. In order to minimize the total squared error, E is differentiated with respect to $\{a_i, b_i\}$ and setting the results to zeros, yields:

$$R\underline{a} = \underline{c} \tag{2-20}$$



7464-80E

FIGURE 2.3.1 BLOCK DIAGRAM OF AN APC SYSTEM WITH ONE LOOP where R is an autocorrelation matrix; $\underline{a}$ and $\underline{c}$ are column vectors defined as:

$$\underline{a} = \begin{bmatrix} a_1 \\ \cdot \\ \cdot \\ \cdot \\ a_p \\ b_1 \\ b_2 \\ b_3 \end{bmatrix} \tag{2-21}$$

and

$$\underline{c} = \begin{bmatrix} r_1 \\ \cdot \\ \cdot \\ \cdot \\ r_p \\ r_M \\ r_{M+1} \\ r_{M+2} \end{bmatrix} \tag{2-22}$$

where r_i is the autocorrelation coefficient of the input speech s_n with that of a delay i. Let the column vector $\underline{d}$ be

$$\underline{d} = \begin{bmatrix} 1 \\ 2 \\ \cdot \\ \cdot \\ \cdot \\ P \\ M-1 \\ M \\ M+1 \end{bmatrix} \tag{2-23}$$

then the elements of the autocorrelation matrix, R, can be expressed as

$$\gamma_{ij} = r_{|d_i - d_j|} \tag{2-24}$$

$$= \sum_{n=1}^{L-|d_i-d_j|} s_n \, s_{n-|d_i-d_j|}$$

In this APC scheme, the autocorrelation coefficients $\{r_i\}$ are needed to define the vector $\underline{c}$ in eq.(2-22) and the autocorrelation matrix R in eq.(2-24). Since the number of the required autocorrelation coefficients is large, their calculation can be best made via the fast Fourier transform technique. Then the calculation of $\{a_i, b_i\}$ can be performed from eq.(2-20) by multiplying the inverse of the matrix R with $\underline{c}$ as follows:

$$\underline{a} = R^{-1}\underline{c} \tag{2-25}$$

The values of the column vector $\underline{a}$ will be used for the computation of the residual signal.

2.3.2 Adaptive Predictive Coder with two Loops

The block diagram of a two-loop APC system is shown in Figure 2.3.2. In this scheme, the predictor loops are divided into P_1 and P_2. Acknowledging the fact that speech signals are often quasi-periodic with period M, the loop P_1 can be used to reduce the redundancy. In particular, if a 3rd order pitch predictor is utilized, the present sample may be estimated as:

$$s_n = \beta_1 S_{n-M+1} + \beta_2 S_{n-M} + \beta_3 S_{n-M-1} \tag{2-26}$$

where β_i's are the pitch prediction coefficients. Third order pitch predictor is best used to compensate the effects of the quantization error in estimating the pitch period. Let the residual error be

$$\begin{aligned} \nu_n &= s_n - \hat{s}_n \qquad (2\text{-}27) \\ &= s_n - \beta_1 S_{n-M+1} - \beta_2 S_{n-M} - \beta_3 S_{n-M-1} \end{aligned}$$

Then, the total squared error may be expressed as:

$$E = \sum_{n=1}^{L} \nu_n^2 \tag{2-28}$$

where L is the number of samples within a frame. In order to minimize the total squared error, differentiating E with respect to β_i and setting the results to zeros, yield:

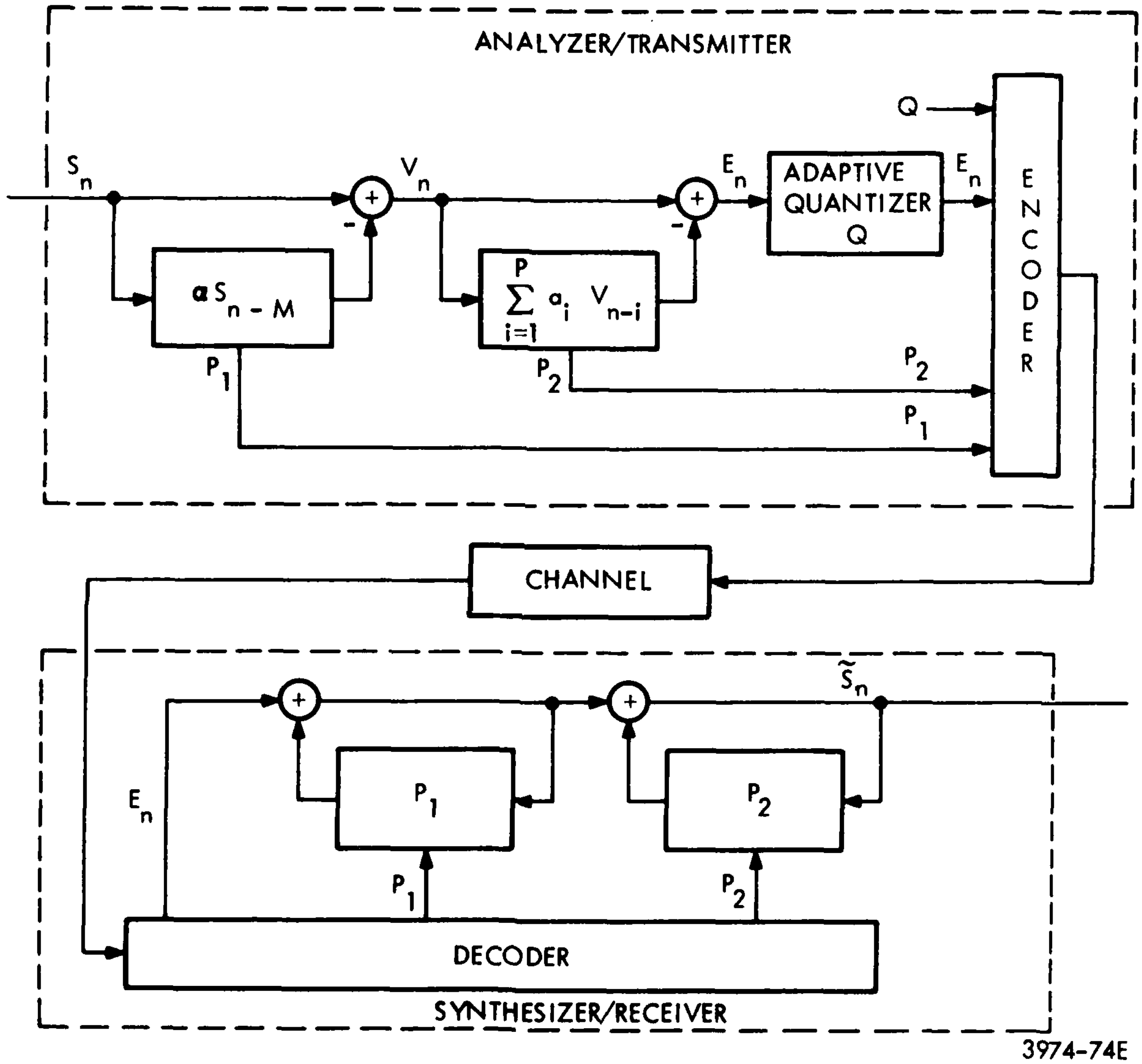


FIGURE 2.3.2: BLOCK DIAGRAM OF AN ADAPTIVE PREDICTIVE CODER WITH TWO LOOPS

$$\begin{bmatrix} \gamma_{11} & \gamma_{12} & \gamma_{13} \\ \gamma_{21} & \gamma_{22} & \gamma_{23} \\ \gamma_{31} & \gamma_{32} & \gamma_{33} \end{bmatrix} \begin{bmatrix} \beta_1 \\ \beta_2 \\ \beta_3 \end{bmatrix} = \begin{bmatrix} \alpha_1 \\ \alpha_2 \\ \alpha_3 \end{bmatrix} \qquad (2\text{-}29)$$

where

$$\alpha_i = \sum_{n=1}^{L} S_n \, S_{n-M+2-i} \quad , \qquad i = 1, 2, 3 \qquad (2\text{-}30)$$

and

$$r_{ij} = \sum_{n=1}^{L} S_{n-M+2-i} \; S_{n-M+2-j} \quad , \qquad i, j = 1, 2, 3 \qquad (2\text{-}31)$$

For quasi-periodic inputs, such as vowels, the magnitude of β_i is large while the magnitude of β_i is near zero for noise-like consonants.

The reduced waveform still contains sufficient redundancy such that a second predictor loop can further reduce the output signal energy, especially if the speech is not periodic or if the pitch period is estimated incorrectly. This second predictor uses a weighted sum of P past samples of the speech waveform to form the estimate as

$$\hat{\nu}_n = \sum_{i=1}^{P} a_i \, \nu_{n-i} \qquad (2\text{-}32)$$

where P is the order of the predictor and $\{a_i\}$'s are chosen to minimize the total squared error

$$U = \sum_{n=1}^{L} (\nu_n - \hat{\nu}_n)^2 \tag{2-33}$$

The predictor coefficients can be obtained by inverting the following matrix equation:

$$\phi \underline{a} = \underline{c} \tag{2-34}$$

where the (i^{th}, j^{th}) element of ϕ is given as:

$$\phi_{ij} = \sum_{n=1}^{L} \nu_{n-i} \nu_{n-j} \tag{2-35}$$

and the i^{th} element of $\underline{c}$ is shown as:

$$C_i = \sum_{n=1}^{L} \nu_n \nu_{n-i} \tag{2-36}$$

If we window the reduced waveform so that it is zero outside the frame interval $1 \leq n \leq L$ (stationary assumption), eq.(2-34) is reduced to the autocorrelation normal equation;

$$\phi_{ij} = \sum_{n=1}^{L-|i-j|} \nu_n \nu_{n-|i-j|} \tag{2-37}$$

$$= \gamma_{|i-j|}$$

and

$$\begin{bmatrix} \gamma_0 & \gamma_1 & \cdot & \cdot & \cdot & \gamma_{p-1} \\ \gamma_1 & \gamma_0 & \cdot & \cdot & \cdot & \gamma_{p-2} \\ \cdot & \cdot & \cdot & \cdot & \cdot & \cdot \\ \cdot & \cdot & \cdot & \cdot & \cdot & \cdot \\ \cdot & \cdot & \cdot & \cdot & \cdot & \cdot \\ \gamma_{p-1} & \gamma_{p-2} & \cdot & \cdot & \cdot & \gamma_0 \end{bmatrix} \begin{bmatrix} a_1 \\ a_2 \\ \cdot \\ \cdot \\ \cdot \\ a_p \end{bmatrix} = \begin{bmatrix} \gamma_1 \\ \gamma_2 \\ \cdot \\ \cdot \\ \cdot \\ \gamma_p \end{bmatrix} \tag{2-38}$$

This is a symmetric Toeplitz matrix because the elements along the principal diagonal and those that are parallel to the diagonal are identical. Efficient solutions exist that result in a_i's that minimize the mean squared energy U in the difference signal. In addition, because the elements of the matrix arise from an autocorrelation function, the stationary matrix solution for the a_i's will yield a stable filter during synthesis, with the recursive filters shown in Figure 2.3.3. Unfortunately, the a_i's are not good transmission parameters because quantization or errors in transmission can cause the poles of the receiver filter given by $1/(1-P_1)(1-P_2)$ to move outside the unit circle in the Z-plane, resulting in unstable waveforms. Consequently, auxiliary parameters called Partial Correlation (PARCOR) coefficients K_i (which are the negatives of reflection coefficients) are calculated from the a_i's, and as long as these PARCOR coefficients have magnitudes less than unity, system stability is assured.

In the actual APC algorithm, instead of quantizing the error signal as depicted in Figure 2.3.2, the analyzer is re-formulated with the quantizer placed inside the filtering loop as shown in Figure 2.3.3. In the absence of the quantizer, this configuration has the same transfer function $(1-P_1)(1-P_2)$ as that of Figure 2.3.2. However, with this new formu-

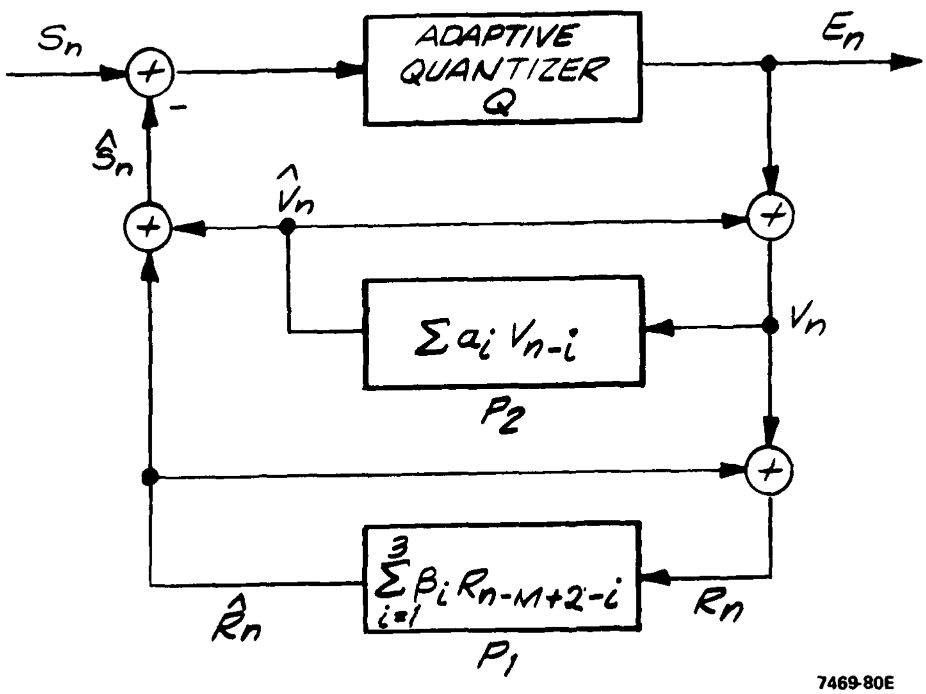


FIGURE 2.3.3 RECONFIGURATION OF THE ANALYZER OF A TWO-LOOP APC TO MINIMIZE THE EFFECT OF QUANTIZATION

lation, it can be shown that the accumulation of quantization errors is eliminated. To illustrate this, the synthesized sample R_n is rewritten as follows:

$$R_n = V_n + \hat{R}_n \tag{2-39}$$

$$= V_n + \hat{S}_n - \hat{V}_n$$

$$= E_n + \hat{S}_n$$

Also, E_n is given by:

$$E_n = S_n - \hat{S}_n + q_n \tag{2-40}$$

where q_n is the quantization noise generated at the nth sample. Substituting eq.(2-40) into (2-39), the following is obtained:

$$R_n = S_n + q_n \tag{2-41}$$

From eq.(2-41), it is clear that the APC synthesized outputs are exactly equal to the inputs except for the quantizing noise q_n. In the case of the APC system with the quantizer outside the predictor loop, the synthesized output R_n is given by:

$$R_n = S_n + \tilde{q}_n \tag{2-42}$$

where $\tilde{q}_n$ is defined as the output obtained when the quantizer noise q_n is fed into the inverse APC filter

$$\tilde{q}_n = q_n + \Sigma\, a_i \tilde{q}_{n-i} \tag{2-43}$$

With the APC system shown in Figure 2.3.2, the synthesized output is an estimate of the input signal plus quantization noise which is an accumulation of previous errors. Hence, with the use of the system shown in Figure 2.3.3, this accumulation effect can be totally avoided.

After computing the pitch gain β_i, the period M, and the PARCOR coefficient K_i, the analyzer digitally filters the speech and quantizes the error signal. Then the K_i, β_i, M, the quantized error sequence e_n, are quantized and sent to the receiver where the predictor coefficients are regenerated iteratively by computing:

$$a_j^{(i)} = a_j^{(i-1)} - a_{i-j}^{(i-1)} * K_i \qquad j = 1, 2, 3, \ldots, i-1 \tag{2-44}$$

$$a_i^{(i)} = K_i \qquad i = 1, 2, 3, \ldots, p \tag{2-45}$$

where $a_j^{(i)}$ represents a_j on the ith iteration.

The synthesizer then creates an output time waveform that is both intelligible and pleasing to listen to. Moreover, this output is reasonably insensitive to errors in pitch extraction, because the P2 predictor on the reduced waveform and the quantization of the error signal can partially compensate for wrong pitch values used in the first predictor P1. In fact, if the pitch period is incorrectly doubled, as it often happens in practice, then predictions made by the first loop P1 are generated from those that are two periods before, and this is not a serious error. If incorrect values for M are chosen, different values of β_i and PARCOR coefficients are also computed to compensate in part for this error. Finally, the error signal, though coarsely quantized, carries considerable information about the true pitch, should this pitch be incorrectly measured. Thus, the APC processed speech does not show severe degradation even in noisy acoustic environments and with many speakers where accurate pitch extraction is difficult.

2.3.3 Tradeoff Analysis of SBAPC Systems

There are several parameters that affect the performance of a SBAPC system for a given transmission data rate, and they are: the sampling rate, the frame size or duration, the number of prediction loops, the order of the short-term predictors, the order of the pitch predictor, quantizer characteristics, bit allocations between side information parameters and residual error signals, and bit allocations between low and high band signals. In this study, a tradeoff analysis has been performed between the number of prediction loops, the order of the short-term predictor, and the order of pitch predictors in each band. Other factors relating to the performance of the SBAPC system have been fixed at values given by: sampling rate = 6400 Hz, frame size or duration = 22.5 msec, number of bits for encoding low-band error signal = 3, and number of bits for encoding high band error signal = 2. These configurations lead to a data rate slightly higher than 16 KBPS. Laplacian quantizers [9, 10] were used for the quantizatization of low and high band error signals until the new quantizer was developed from the actual distribution of the error signal's amplitudes. Quantizations were not applied to the parameters of side information, since the first objective of this tradeoff analysis was to determine the number of loops and the order of the predictor in each loop. The quantizations, bit allocations to the side information, the residual error signals, and forward error correcting code will be considered in the later sections.

2.3.3.1 The Performance of the SBAPC System with one Loop

The block diagram of a SBAPC system with one loop is shown in Figure 2.3.4. In this scheme, only short-term predictors are considered in the high band prediction loop, since the signals of the high band often contain little information of pitch. The performance in terms of the signal-to-quantization noise ratio (S/Q) is tabulated in Table 2.2. In this experiment, the data rate used for encoding the error signal is 16 Kb/s. While the total data with side information is higher than 16 Kb/s, the purpose is to study the tradeoffs between the pitch predictor and various predictor orders of the high and low bands. As it is noted from this table, the signal-to-quantization noise ratio (S/Q) is high (about 20 dB). Consequently, the quality of the synthesized speech is very high. The performance of this scheme is better with a larger order of short-term predictors.

A typical S/Q plot of the 1-loop SBAPC systems with and without the pitch predictor are shown in Figure 2.3.5. As it is noted from this figure, the performance is insensitive with respect to the presence of the pitch predictor. Only one pitch related predictor is considered in the evaluation of performance since the system is often unstable for large order (≥ 2) of pitch related predictors. Also, the S/Q from Table 2.2 indicates that small number of prediction coefficients ($\simeq 4$) is preferable since the performance does not improve significantly when more than 4 prediction coefficients are used.

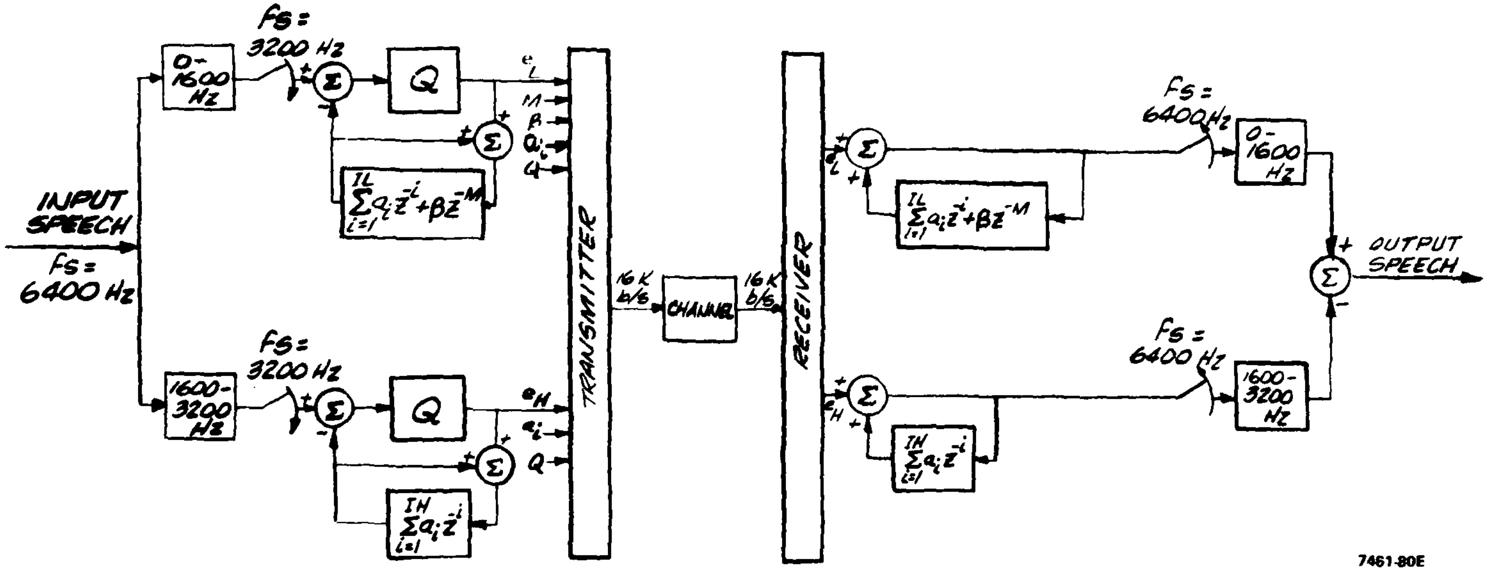


FIGURE 2.3.4 BLOCK DIAGRAM OF THE SBAPC SYSTEM WITH ONE LOOP

LB Order \ HB Order	2	4	6
4	20.14 dB +	20.43 +	20.47 +
	19.83 *	20.12 *	20.17 *
6	20.37 +	20.66 +	20.70 +
	20.19 *	20.48 *	20.53 *
8	20.55 +	20.87	20.92 +
	20.31 *	20.61 *	20.65 *

+ with one pitch predictor
* with no pitch predictor

HB = High Band

LB = Low Band

Table 2.2: Signal-to-Quantization Noise Ratios of the SBAPC System with one loop at 16 KBPS

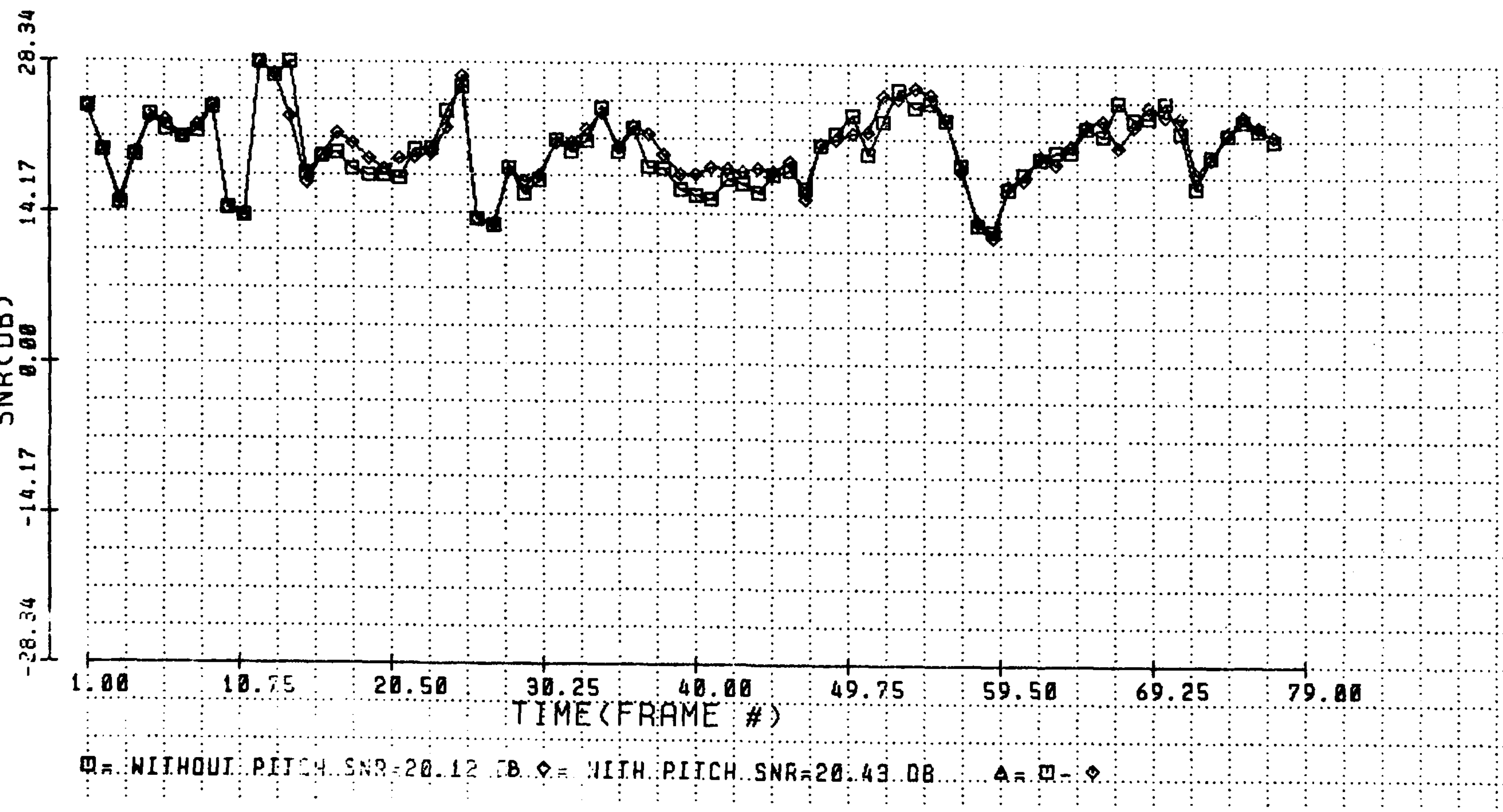


FIGURE 2.3.5: A SIGNAL-TO-QUANTIZATION NOISE RATIO PLOT OF THE 16 KBPS SBAPC SYSTEM WITH ONE LOOP

2.3.3.2 The Performance of a SBAPC System with two Loops

The block diagram of a SBAPC system with two loops is shown in Figure 2.3.6. In this scheme, only short-term predictors (one loop) are considered in the high band since signals of the high band after downsampling often contain little information about pitch. The performance in terms of S/Q are tabulated in Table 2.3 with the order of predictors as a variable. The S/Q of the SBAPC system with two loops increases as the order of the low-band predictors increases as shown in Table 2.3. However, the improvement is not that dramatic with the increase in higher band predictors. The SBAPC system with one loop (when the pitch information is not used) performs as well as this scheme for unvoiced frames. However, the SBPAC system with two loops always performs better than the system with one loop for voiced speech. In this scheme, the first order pitch loop results in an increase of 2-3 dB of S/Q over the scheme with no pitch predictor. An additional increase of 1-2 dB of S/Q may be obtained if three pitch predictors are used, but the system is sometimes unstable at low data rates. At 16 KBPS, these distortions are not noticeable in informal listening tests. As the data rate decreases to 9.6 KBPS, the lowering in S/Q becomes more perceptable. Especially noticeable is the hissing noise at high frequencies when no pitch loop was used. This noise degrades much of the speech quality at data rates below 9.6 KBPS.

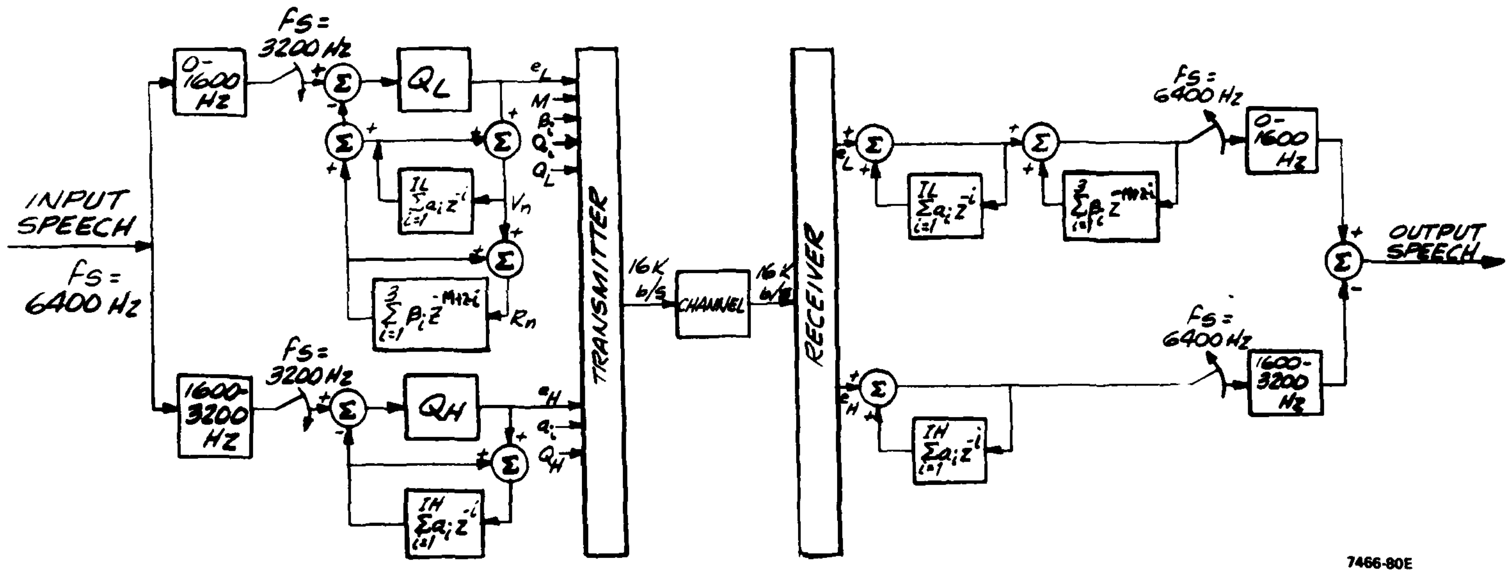


FIGURE 2.3.6 BLOCK DIAGRAM OF A SBAPC SYSTEM WITH TWO LOOPS

HB Order / LB Order	2	4	6
4	22.25 dB+	22.59+	22.66+
	23.46 *	23.77 *	23.83 *
6	22.49 +	22.86+	22.94 +
	23.58 *	23.91 *	23.97 *
8	22.81 +	23.19+	23.26 +
	23.72 *	24.04 *	24.11 *

+ with one pitch predictor
* with three pitch predictors

Table 2.3 Signal-to Quantization Noise Ratio of a SBAPC System with two loops at 16 KBPS

2.4 The Shaping of Quantization Noise in SBAPC Systems

The split-band adaptive predictive coding technique has been proven to be an efficient method for encoding speech signals at 16 Kbps. Though the SBAPC system attempts to minimize the rms value of the error signals in the coded speech, low amplitude quantization error, however, does not always ensure perceptually small distortion in the processed speech. It has been suggested that higher quality speech can be obtained by the adjustment of the noise spectral shape without changing the rms value of the error signals. In the next section, two techniques of noise shaping will be considered [11, 12]. The first method, proposed by Makhoul, changes the flat spectrum of the quantization noise to resemble that of the input speech, whereas the second technique, originated by Atal, attempts to flatten the noise spectrum. Though the objective of both techniques is to enhance the quality of the synthesized speech without additional overhead bits; the complexity, stability of the algorithms, and the flexibility of the noise spectral shaping are quite different.

2.4.1 Makhoul's Noise Shaping Technique

The transmitter portion of the basic APC system is shown in Figure 2.4.1. In this figure, S(z) represents the z transformation of the input speech s_n, n=1, ... L where L is the number of samples in one frame interval. C(z) represents the transfer function of the pitch prediction loop which is given by:

$$C(z) = 1 - \alpha z^{-M} \qquad (2\text{-}46)$$

where M is the number of speech samples in one pitch period and α is the pitch gain parameter. α is computed in order to minimize the total

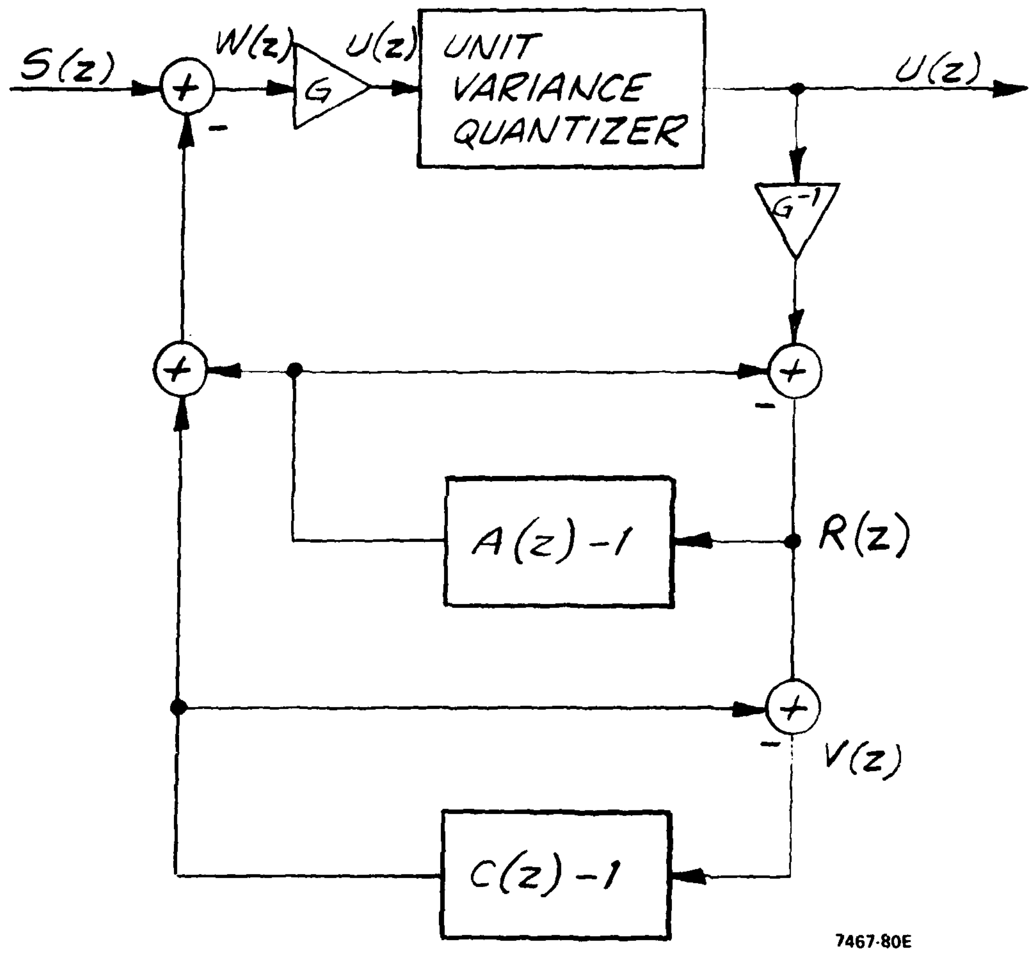


FIGURE 2.4.1 BLOCK DIAGRAM OF APC SYSTEM

$$\text{squared error } E = \sum_{n=1}^{L} (S_n - S_{n-M})^2 \tag{2-47}$$

and it can be expressed as:

$$\alpha = \frac{\sum_{n=1}^{L} S_n S_{n-M}}{\sum_{n=1}^{L} S_{n-M}^2} \tag{2-48}$$

The transfer function of the short-term prediction can be written as:

$$A(z) = 1 + \sum_{k=1}^{P} a_k z^{-k} \tag{2-49}$$

where $\{a_k\}$'s are the linear prediction coefficients. $\hat{W}(z)$ represents the z transform of the APC residual signal which includes the effects of the quantization noise $Q(z)$; i.e.,

$$Q(z) = \hat{W}(z) - W(z) \tag{2-50}$$

where $W(z)$ is the z transform of the APC residual signal only. $U(z)$ is the residual error signal with unit variance and can be written as:

$$U(z) = G\,W(z) \tag{2-51}$$

where G is a normalization gain factor. The spectrum of $W(z)$ is assumed to be flat which is a reasonable assumption, since the residual error appears to be highly uncorrelated. Thus, the input of the quantizer is assumed to be white and has unit variance. At the receiver, $U(Z)$, after multiplying G^{-1}, is sent through the all-pole linear prediction filter $1/A(z)$, and the pitch prediction filter $1/C(z)$. The output signal at the receiver is exactly equal to the signal $V(z)$ for voiced speech or $R(z)$ for unvoiced speech, provided the digital data transmission is error-free.

The main idea in noise shaping is to employ a linear filter and modify the quantization noise according to a pre-determined perceptual criterion. For simplicity reasons, the analysis will be conducted without the pitch loop. In this manner, the reconstructed signal $R(Z)$ is given by:

$$R(Z) = S(Z) + B(Z)Q(Z) \tag{2-52}$$

where $B(Z)$ is the z-transform of the shapinq filter. Since $R(Z)$, as shown in Figure 2.4.1, is also equal to:

$$R(Z) = \frac{\hat{W}(Z)}{A(Z)} = \Big(W(Z) + Q(Z)\Big)/A(Z) \tag{2-53}$$

Equating Eq. (2-53) with (2-52), the result becomes:

$$W(Z) = A(Z)S(Z) + \Big[A(Z)B(Z) - 1\Big]Q(Z) \tag{2-54}$$

Adding and subtracting $S(Z) + B(Z)Q(Z)$, Eq. (2-54), can be rewritten as:

$$W(Z) = S(Z) + \Big(B(Z) - 1\Big)\,Q(Z) + \Big(A(Z) - 1\Big)\Big(S(Z) + B(Z)Q(Z)\Big) \tag{2-55}$$

Substituting Eqs. (2-52) and (2-53) into (2-55), the following is obtained:

$$W(Z) = S(Z) + \Big(B(Z) - 1\Big)Q(Z) + \Big(A(Z) - 1\Big)\Big(W(Z)/A(Z)\Big) \tag{2-56}$$

The noise shaping algorithm, as illustrated in Eq. (2-56) is depicted in Figure 2.4.2.

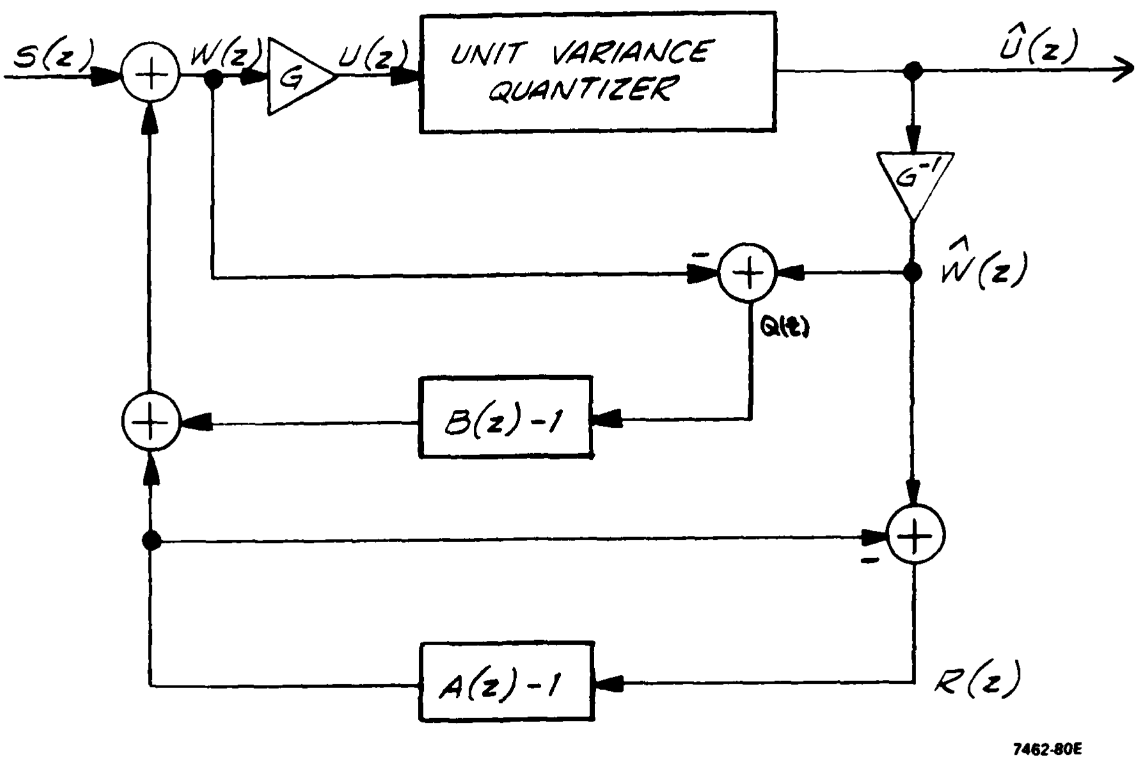


FIGURE 2.4.2 BLOCK DIAGRAM OF THE APC SYSTEM WITH NOISE SHAPING

In this study, the simple and stable second order all-zero filter is employed as B(Z). Its transfer function, given as $B(Z) = \sum_{i=0}^{2} b_i z^{-i}$, is estimated from the original transfer function of the predictor coefficients A(Z) as follows:

$$\rho_i = \sum_{k=0}^{p-|i|} a(k)\, a(k + |i|) \qquad 0 \leq i \leq 2 \tag{2-57}$$

where p is the order of the filter A(Z). The coefficients b_n are computed from the set of linear normal equations:

$$\sum_{n=1}^{2} b_n \rho_{i-n} = -\rho_i \qquad 1 \leq i \leq 2 \tag{2-58}$$

and this results in:

$$\begin{aligned} b_0 &= 1 \\ b_1 &= \rho_1(\rho_2 - \rho_0)/(\rho_0^2 - \rho_1^2) \\ b_2 &= (\rho_1^2 - \rho_0\rho_2)/(\rho_0^2 - \rho_1^2) \end{aligned} \tag{2-59}$$

To study the effect of noise shaping on the SBAPC system, the 2nd order all-zero filter, as given in Eq. (2-59), has been incorporated. In particular, the performance of the combined system at the data rate 8 Kbps (quantization of error signal only, excluding side information) has been investigated. The results in terms of S/Q are tabulated in Table 2.4 for the SBAPC system with and without noise shaping on each band. As it is noted in Table 2.4, maximum signal-to-quantization noise ratio is achieved with the absence of noise shaping on both bands, but the output speech obtained without noise shaping exhibits a rough quality together with some disturbing

noise shaping	number of pitch predictor 1	number of pitch predictor 3
lowband: yes highband: yes	14.38 dB	15.55 dB
lowband: yes highband: no	14.43 dB	15.59 dB
lowband: no highband: no	15.43 dB	16.84 dB

Table 2.4 SIGNAL TO NOISE RATIO OF SBAPC SYSTEM WITH MAKHOUL'S NOISE SHAPING AT 8 KBPS

background noise. However, these noises disappear when noise shaping is applied. Informal listening tests further suggest that noise shaping in the high band does not enhance the subjective speech quality for male speakers. However, more careful listening tests reveal that the "beeping" noise which occurs frequently with female speakers at low data rates (e.g., 8 Kb/s) is reduced when noise shaping is applied to both high and low bands. For higher data rate systems (>16 Kb/s), noise shaping is not really helpful.

2.4.2 Atal's Noise Shaping Technique

The block diagram of a generalized adaptive predictive coder with adjustable noise spectrum is shown in Figure 2.4.3. In this figure, the speech samples $\{s_n\}$ pass through the inverse linear prediction filter, $1 - A(z)$, where

$$A(z) = \sum_{k=1}^{m} a_k z^{-k} \qquad (2\text{-}60)$$

and m is the order of the filter and $\{a_k\}$'s are the linear prediction filter coefficients. The output of the filter may be expressed as

$$d_n = s_n - \sum_{k=1}^{m} a_k s_{n-k} \qquad (2\text{-}61)$$

Then d_n is fed into the quantizer loop. For the sake of simplicity, the pitch loop is ignored. Then, the quantizer noise δ_n defined as the difference between the quantizer output $\hat{q}_n$ and the quantizer input q_n may be expressed as

$$\delta_n = \hat{q}_n - q_n \qquad (2\text{-}62)$$

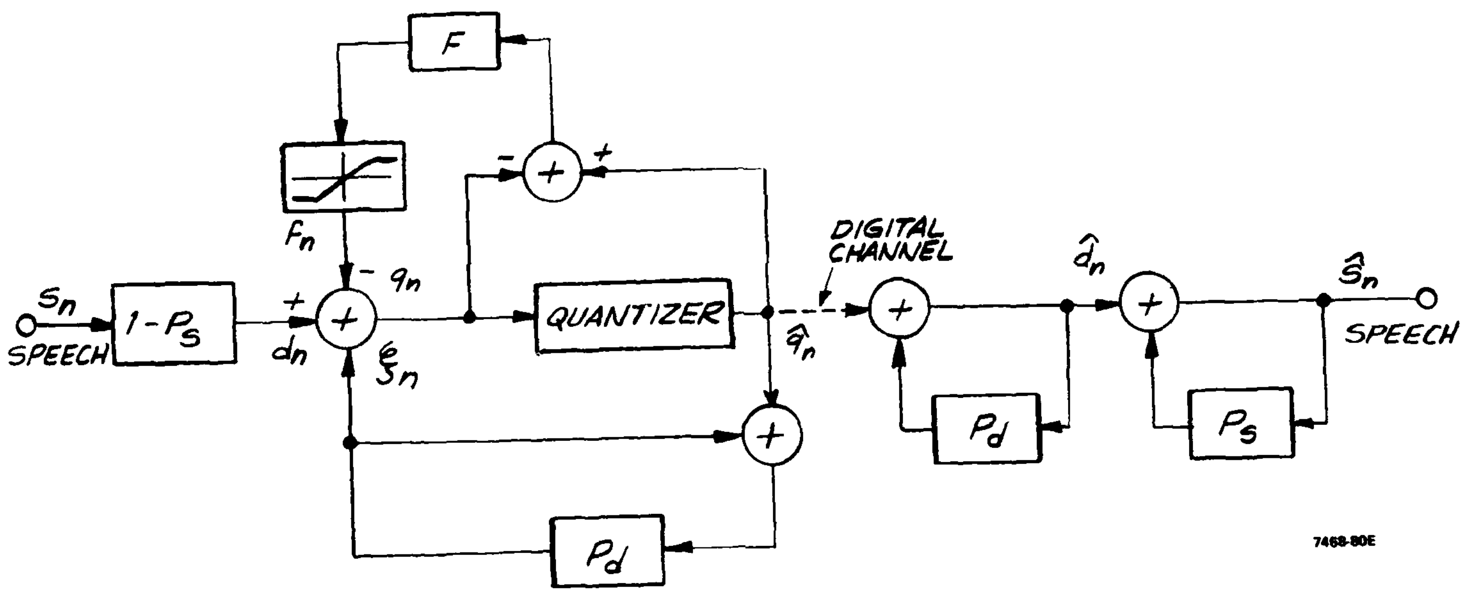


FIGURE 2.4.3 BLOCK DIAGRAM OF THE GENERALIZED APC SYSTEM WITH ATAL'S NOISE SHAPING TECHNIQUE

The quantizer noise δ_n is fed into the noise shaping filter F(z), and the output of this filter f_n is subtracted from d_n to yield the quantization noise as:

$$q_n = d_n - f_n$$

$$= s_n - \sum_{k=1}^{m} a_k s_{n-k} - \sum_{k=1}^{m'} b_k \delta_{n-k} \tag{2-63}$$

where m' is the order of the feedback loop noise shaping filter and $\{b_k\}$'s are the coefficients of the filter F(z) defined as:

$$F(z) = \sum_{k=1}^{m'} b_k z^{-k} \tag{2-64}$$

The output of the predictive coder can now be expressed as:

$$\hat{s}_n = \hat{q}_n + \xi_n$$

$$= \hat{q}_n + \sum_{k=1}^{m} a_k \hat{s}_{n-k}$$

$$= q_n + \delta_n + \sum_{k=1}^{m} a_k \hat{s}_{n-k}$$

$$= s_n + \sum_{k=1}^{m} a_k(\hat{s}_{n-k} - s_{n-k}) + \delta_n - \sum_{k=1}^{m'} b_k \delta_{n-k} \tag{2-65}$$

Taking the z transformation of this equation yields

$$\hat{S}(z) - S(z) = \Delta(z) \frac{1 - F(z)}{1 - A(z)} \tag{2-66}$$

where $\Delta(z)$, $\hat{S}(z)$, $S(z)$ represent the z transform of δ_n, $\hat{s}_n$, and s_n, respectively. The total processing noise will be the same as the quantizer noise $\Delta(z)$, if $F(z)$ equals $A(z)$. In other words, the spectrum of the output noise can be controlled with great flexibility with the feedback filter $F(z)$. Assuming that the power of the quantizer noise δ_n does not vary significantly by the feedback loop $F(z)$, the average value of the power spectrum of the output noise is determined only by the quantizer and is not altered by the choice of $F(Z)$ or $A(Z)$. However, the filter, $F(z)$, distributes the noise power from one frequency to another. Thus, reduction of quantizer noise at one frequency can be obtained at the expense of increasing the quantizer noise of another one. Since a large part of the perceived noise in a coder comes from the frequency regions where the signal is low, the filter $F(z)$ may be used to reduce the noise in such regions while increasing the noise in the formant regions where the noise could be effectively masked by the speech signal. It is also assumed that the quantizer noise is uncorrelated with the prediction error signal, which is a reasonable assumption, particularly when the prediction error signal is white. Then the power of the quantizer may be expressed as

$$E_q = E_p + E_f \tag{2-67}$$

where E_p is the power of the prediction error signal and E_f is the power of the filtered quantized noise. In many cases, it is desirable to have a small E_f to ensure the small changes of the quantized noise. Atal suggested that the APC system of Figure 2.4.3 can be operated in stable condition by adding a high passed filtered noise to the input speech. Consequently, the terms in the covariance matrix and the correlation vector have to be modified as:

$$\hat{\gamma}_{ij} = \gamma_{ij} + \lambda\, E_{min}\, \mu_{i-j} \tag{2-68}$$

and

$$\hat{c}_i = c_i + \lambda\, E_{min}\, \mu_i \tag{2-69}$$

where

$$\gamma_{ij} = \left\langle s_{n-i}\, s_{n-j} \right\rangle \tag{2-70}$$

$$c_i = \left\langle s_n\, s_{n-i} \right\rangle \tag{2-71}$$

The λ in eq. (2-68) is a small constant in the range 0.01-0.1, E_{min} is the minimum value of the mean squared prediction error, μ_i is the autocorrelation of the high-pass filtered white noise at a delay of i samples, and $\langle \cdot \rangle$ denotes time averaging. In this study, μ_i's are chosen to be $\mu_0 = 3/8$, $\mu_1 = 1/4$, $\mu_2 = 1/16$, $\mu_k = 0$ for $k \geq 3$. With the addition of the high passed filtered noise, the stability of the feedback loop filter is increased.

The quantizer input $\{q_n\}$ in Figure 2.4.3 sometimes contain large amplitudes especially for periodic signals in voiced speech. However, the large amplitudes can be removed by pitch prediction. The transfer function of this pitch loop may be expressed as:

$$P_d(z) = \beta_1 z^{-M+1} + \beta_2 z^{-M} + \beta_3 z^{-M-1} \tag{2-72}$$

where M represents the number of samples in one pitch period and β_1, β_2, β_3 are the filter coefficients in the pitch filter. The values of β_1, β_2, β_3

may be obtained from the set of simultaneous linear equations, which is described in section 2.3.2.

The addition of the high pass filtered noise to the speech input increases the stability of the quantization feedback loop. A simple and effective solution to ensure the stability of the feedback loop is to limit the peak amplitude of the sample f_n as shown in Figure 2.4.3. The appropriate peak limiter in the feedback loop limits the samples $\{f_n\}$ to a maximum value of twice the rms value of the prediction error. For some choices of F(z), several instances of instability in the feedback loop have been encountered without peak limiter. However, the inclusion of the peak limiter in the feedback loop removes instability in those frames, and it does not increase the quantization noise significantly. Atal suggested that the feedback loop filter shown as

$$F(z) = A(\alpha z^{-1}), \qquad 0 \leq \alpha \leq 1 \tag{2-73}$$

is a good choice because of the simplicity and flexibility of controlling the spectral shape of the quantization noise.

The above noise shaping method was applied to the SBAPC algorithm. Several choices of the feedback loop filter F(z) were considered in each band and their S/Q's are tabulated in Table 2.5. For $\alpha_\ell = \alpha_h = 0$ (or $F(Z) = 0$ in both low and high bands), the quantization noise has the same spectral envelope as the original speech as illustrated in the spectral plot of 1 frame of data in Figure 2.4.4. This particular choice of α's leads to a noisy output of 11.61 dB S/Q at 8 Kb/s (excluding side information). As the value of α increases, the performance of the SBAPC scheme in terms of S/Q improves. The ratio tends to reach the maximum

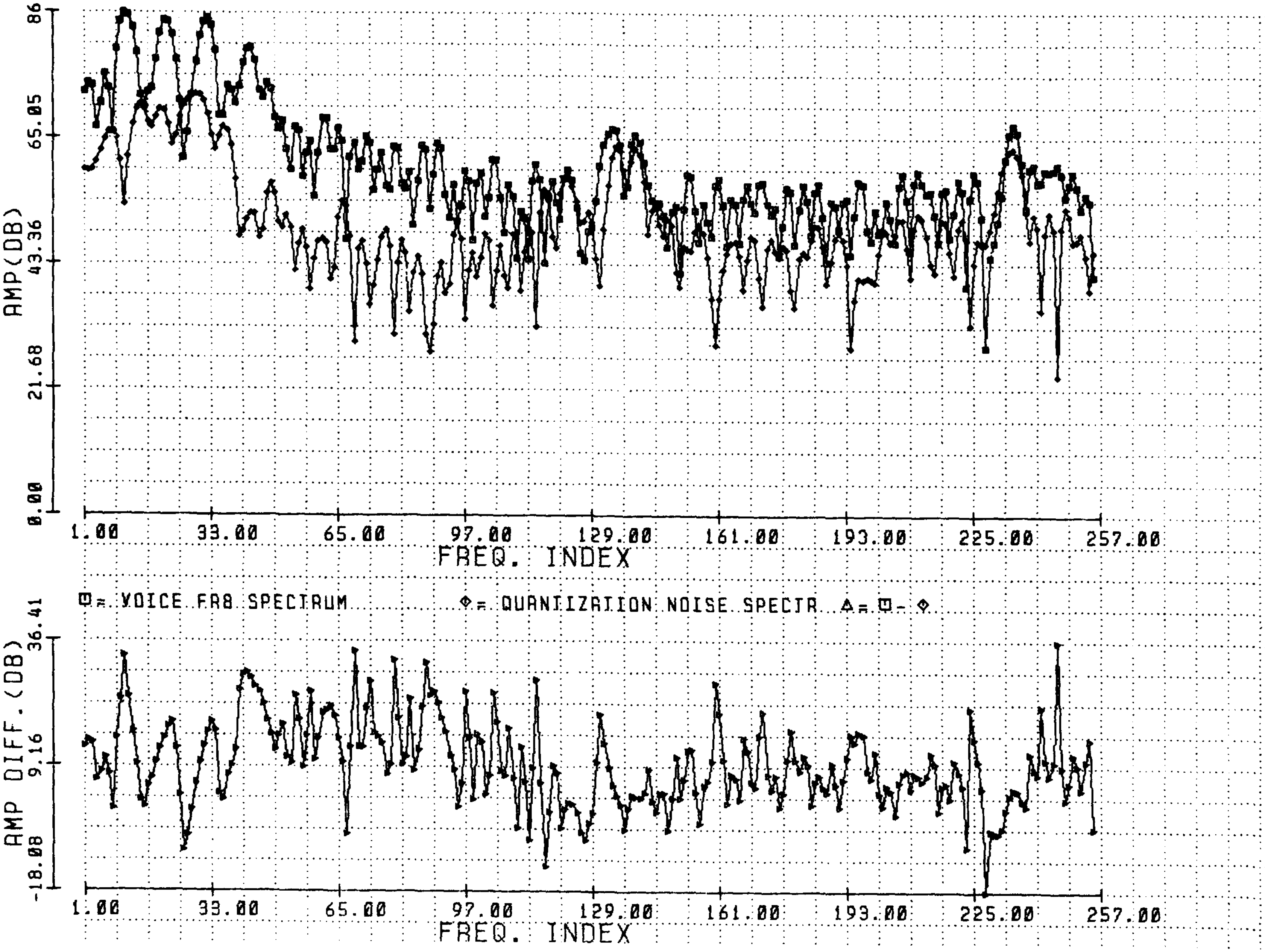
α in LB \ α in HB	0.0	0.25	0.5	0.75	1.0
0.0	11.61 dB	11.66	11.69	11.73	11.77
0.4	13.04	13.09	13.12	13.17	13.22
0.5	14.33	14.39	14.44	14.49	14.55
0.75	15.82	15.91	15.95	16.05	16.12
1.0	15.56	15.65	15.70	15.81	15.88

HB: Highband

LB: Lowband

Table 2.5 The Signal-to-Quantization Noise Ratio in dB of the SBAPC System with Atal's Noise Shaping at 8 KBPS.

Figure 2.4.4 SPECTRUM PLOT OF THE SBAPC SYSTEM WITH TIME DOMAIN SNR=12.77 DB WITH ALPA=0.0 TO BOTH BAND IN ATAL'S NOISE SHAPING TECHNIQUE



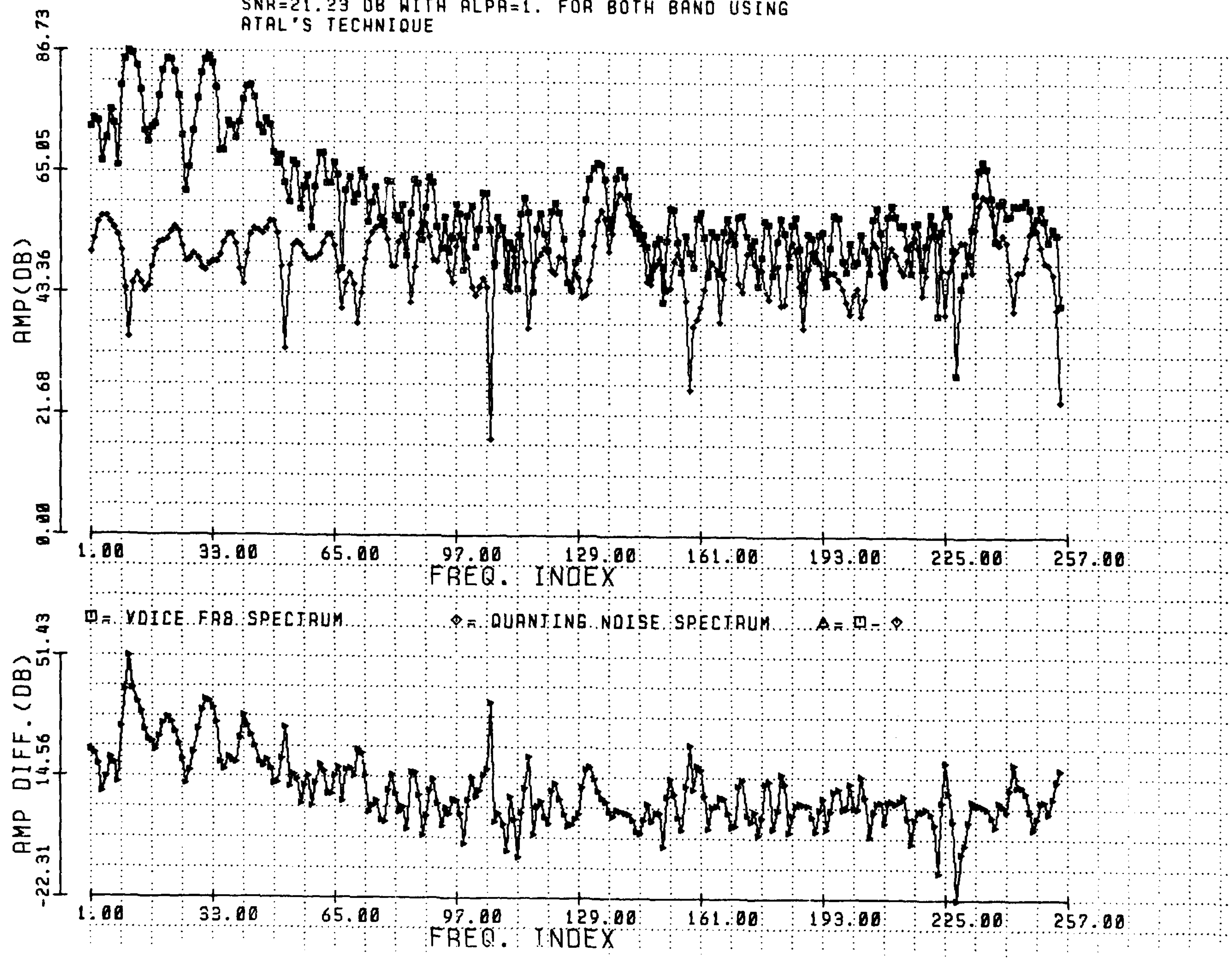
183

for $\alpha_\ell = 0.75$, $\alpha_h = 1.0$. For $\alpha_\ell = \alpha_h = 1$, the system is similar to Makhoul's model shown in Figure 2.4.2 without noise shaping. In this case, the spectrum of the output noise is almost uniform as shown in Figure 2.4.5 and the S/Q is high (15.88 dB). The power of the quantization noise is small, but the reconstructed speech contains much wideband noise For $\alpha_\ell = \alpha_h = 0.75$, the power of the quantization noise is small, and informal subjective listening tests indicate that the choice of $\alpha_\ell = \alpha_h = 0.75$ leads to the highest quality of synthesized speech. It is, therefore, concluded that noise shaping actually enhances the synthesized speech quality of the SBAPC system.

2.4.3 Comparison of Atal's and Makhoul's Noise Shaping Methods

The shaping of quantization noise has been applied to the SBAPC system in order to enhance the quality of synthesized speech at low data rates. Makhoul's technique and Atal's technique have been incorporated in the SBAPC system at the data rate 8 Kbps (quantization of error signal only, excluding side information) in order to determine the effects of noise shaping clearly. The second order all-zero filter proposed by Makhoul has been applied to various input speech signals. The performance of this scheme is tabulated in Table 2.4 in terms of the signal to noise ratio with male input speech. Informal listening tests suggest that noise shaping in the high band does not enhance the subjective speech quality for male speakers. Atal's noise shaping technique has also been applied to the SBAPC system. The system performance is tabulated in Table 2.5 with the parameter α which controls the bandwidth of the noise feedback loop. Informal listening tests suggest that the choice of $\alpha \approx 0.75$ in lowband and highband provides the best subjective speech quality.

Figure 2.4.5 SPECTRUM PLOT OF THE SBAPC SYSTEM WITH TIME DOMAIN SNR=21.23 DB WITH ALPA=1. FOR BOTH BAND USING ATAL'S TECHNIQUE



'10

Both Atal's technique and Makhoul's technique improve the subjective speech quality at the data rates below 12 Kbps. The technique of Makhoul attempts to change the flat spectral shape of the noise toward the spectral shape of input speech while the technique of Atal attempts to change the spectrum of the noise (same shape as input speech) toward the flat spectrum. The goal of both techniques is the same, but the complexity and stability of the SBAPC system and the flexibility of the noise spectral shapes are different. For the purpose of comparison, a typical performance of the SBAPC system using both techniques is tabulated in Table 2.6. Though the signal to quantization noise ratios are approximately equal, the subjective speech quality differs. Informal listening tests suggest that Makhoul's technique provides a slightly better speech quality as compared to that of the Atal technique.

The speech quality of the SBAPC system is quite natural at the data rate 16 KBPS. However, careful informal listening tests indicate that "beeps" are sometimes heard, especially at the low data rates. In general, "beeps" occur more often in female speech than in male speech. Experiments have been conducted to identify the source of the "beeps," and we conclude that it is caused by the coarse quantization of error signal in the highband, since the "beeps" disappear when the signals of the highband are not quantized. This is no easy way to reduce these "beeps." The only alternative is to mask the "beeps" by employing noise shaping at the high band and by inserting a small level random noise at the synthesizer. The random noise is added only when the number of bits allocated to the high band is small (≤ 1). Unfortunately, the amplitude of the random noise has to be adjusted so that the output speech will not be too noisy.

	First order pitch prediction	Third order pitch prediction
Makhoul's technique	17.87 dB	18.94 dB
Atals technique	18.04 dB	18.70 dB

TABLE 2.6: COMPARISON OF SBAPC SYSTEMS WITH TWO DIFFERENT NOISE SHAPING TECHNIQUES

2.5 Quantization of Residual Signals

2.5.1 Adaptive Bit Allocations

One of the advantages of SBAPC algorithms is the possibility of using spectral densities of the subband signals and applying nonuniform quantization to each band. The conventional approach is to use a fixed bit allocation rule where a pre-determined number of bits is assigned to the coding of each band. In particular, a larger portion of the available bits will be allocated for the quantization of the low band in order to capture all pitch and formant information, whereas a smaller portion of the bits will be utilized for the high band. However, since the ratio of low band and high band energies fluctuates from frame to frame, fixed bit allocation may not necessarily be the best strategy. Instead, an adaptive bit allocation scheme that dynamically alters the bit assignments depending on the energies of the two bands seems more applicable.

It has been shown that the most dominant source of speech distortion in the SBAPC system is the quantization of residual waveforms in the prediction loops. To illustrate the effects of quantization errors, the S/Q of the SBAPC system are plotted in Figure 2.5.1 with or without applying quantization to low band and high band error signals. In this figure, the plot of (□) indicates that the quantization is not applied to the low band error signal, and the error signal in the high band is quantized with 2 bits per sample. The plot of (◇) indicates that the quantization is not applied to the high band error signals and the error signals of low band are quantized with 3 bits per sample. In this figure, the SBAPC system performs poorly in regions I1, I2, and I3 as compared to that of (◇).

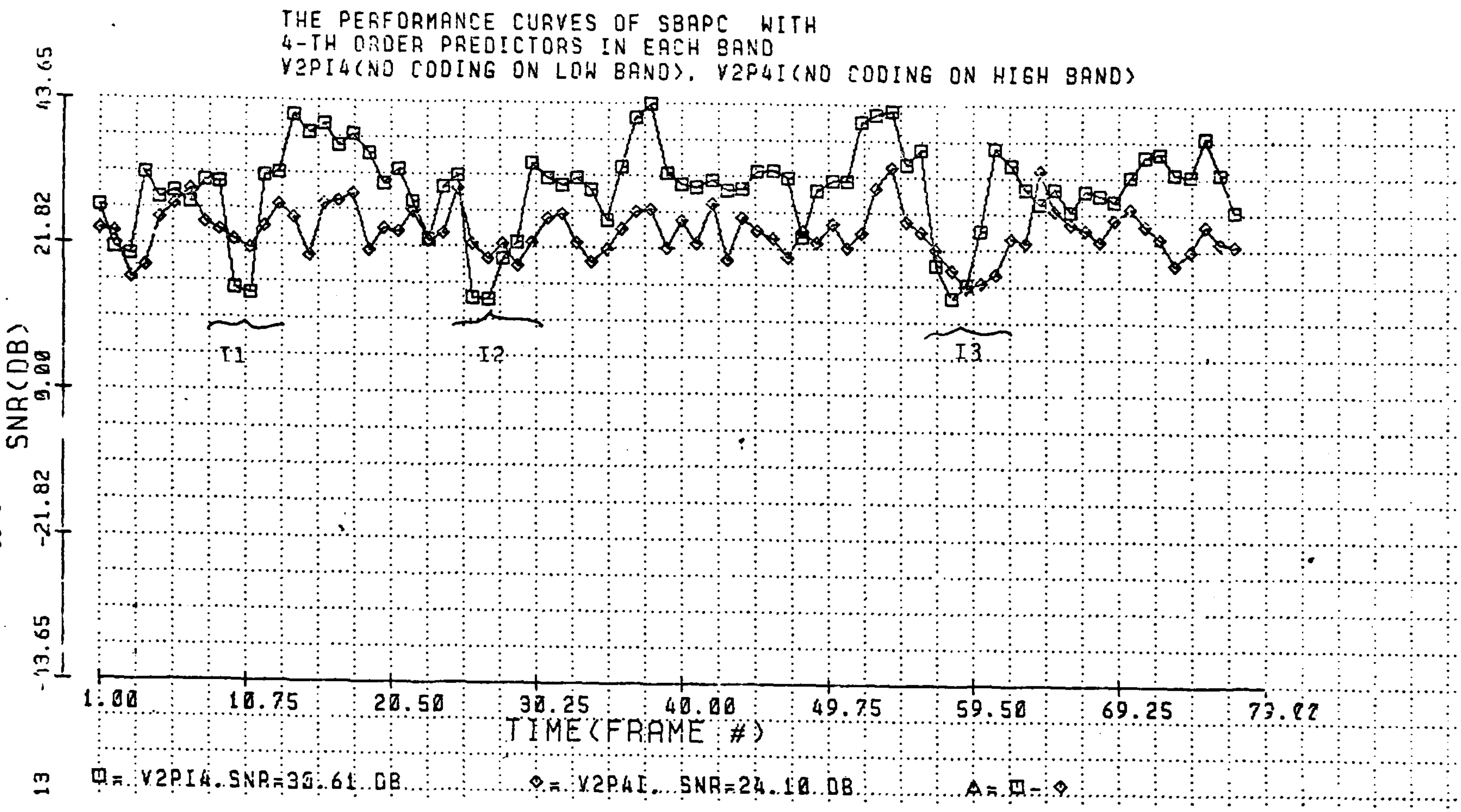


Figure 2.5.1 The Performance Curves of a SBAPC System with or without Quantization of Error Signals

It is, therefore, suggested that more bits may be allocated for the high band error signals in regions I1, I2, and I3 in order to improve the performance with fixed transmission data rate.

The performance of the SBAPC system with or without adaptive bits allocations is then compared and the results are shown in Table 2.7. The SBAPC system works better when the bits employed to quantize the low band and high bands are adaptively allocated according to the energy of each band. The adaptive bit allocations yield an increase of 1 dB in signal-to-quantization noise ratio over the scheme with fixed bit allocations. Since the energies of both bands have to be sent to the receiver, adaptive bit allocations do not require additional overhead bits. However, an additional bit assignment rule may be required in order to avoid the difficulties of encoding signals with non-integer number of bits per sample.

Let R_i be the actual number of bits allocated to the i-th subband, then the average number of bits per band, $\overline{R}$, may be expressed as

$$\overline{R} = \tfrac{1}{2} \sum_{i=1}^{2} R_i \tag{2-74}$$

Let σ_i^2 be the energy of the i-th band, then it can be shown that the optimum bit allocation can be obtained (in the sense of minimizing the rms error of the coded speech) as

$$R_i = \overline{R} + \tfrac{1}{2}\log_2 \frac{\sigma_i^2}{\sqrt{(\sigma_1^2\ \sigma_2^2)}} \ ; \quad i = 1, 2 \tag{2-75}$$

Hence, bits to encode the error signal of each band can be adaptively allocated according to the energy of each band.

HB Order / LB Order	2	4	6
4	22.25 dB	22.59	22.66
	23.28 *	23.[illegible] *	24.11 *
6	22.49	22.86	22.94
	23.66 *	24.07 *	24.23 *
8	22.81	23.19	23.26
	23.64 *	24.24 *	24.25 *

* with adaptive bit allocations

Table 2.7 Signal-to-Quantization Noise Ratio of the two-loop SBAPC System with and without adaptive bit allocations at 16 KBPS

The SBAPC performs better when the bits used to quantize the low band and high band error signals are adaptively allocated according to the energy of each band as shown in Figure 2.5.2. The adaptive bit allocations offer an inverse of 1-2 dB in S/Q over that of a scheme with fixed bit allocations. Only assignments of integer bits per sample are utilized in order to simplify the algorithm. Since the energies of both bands have to be sent to the receiver for quantization purposes, adaptive bit allocations do not require additional transmission of data.

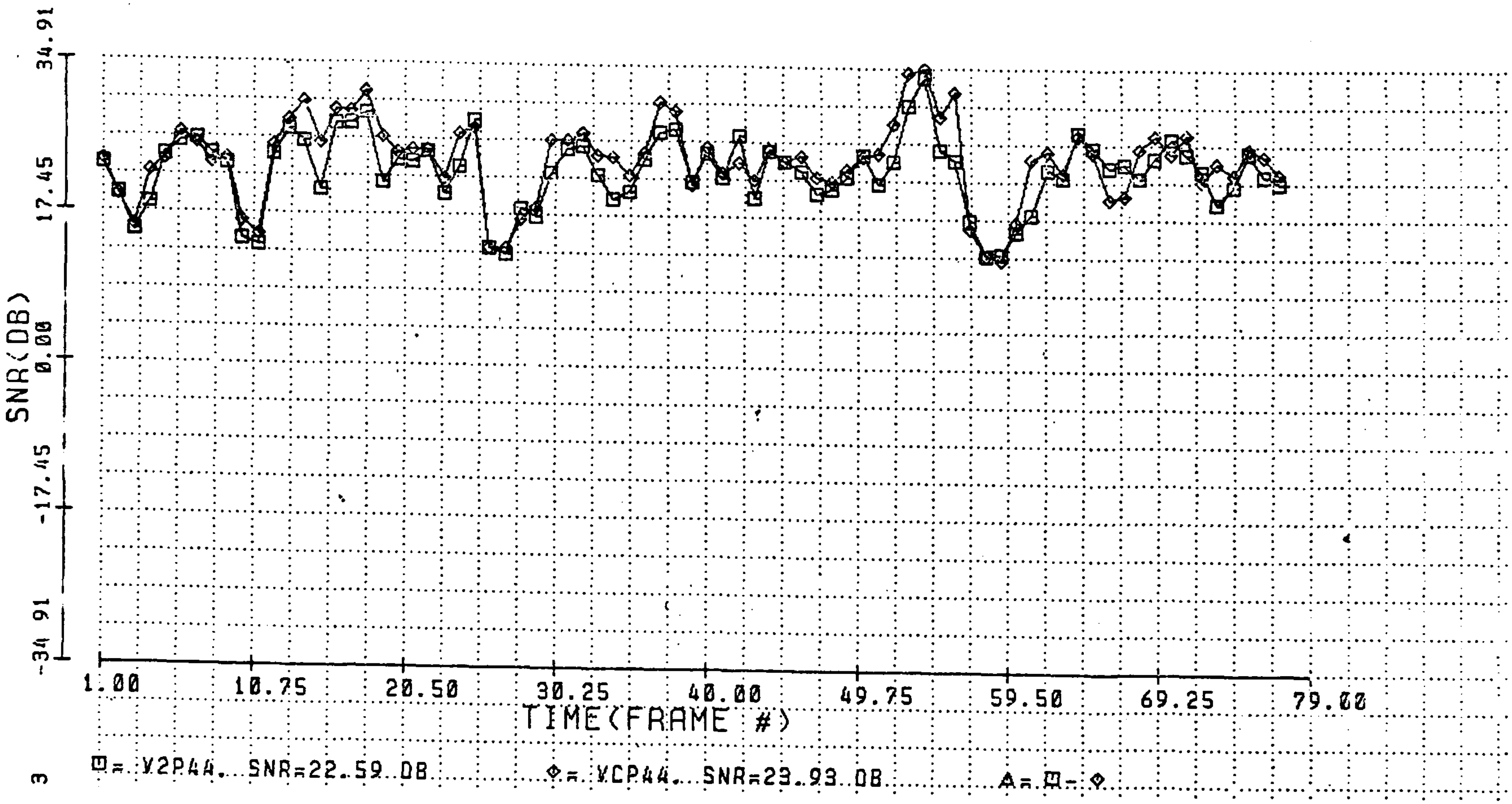


FIGURE 2.5.2 THE PERFORMANCE PLOTS OF 16 KBPS SBAPC SYSTEMS WITH FIXED AND ADAPTIVE BIT ALLOCATIONS

2.5.2 Quantization of Residual Signals

In APC systems, it is well known that the design of residual signal quantizers will significantly affect the voice quality of the processed speech. The SBAPC algorithm, which is very similar to the APC, is of no exception and fine quantization on both high and low band residual signals are vital to its success. In light of the fact that the adaptive bit allocation scheme as discussed in Section 2.5.1 only results in integer bit assignments, the emphasis of this study has been on the designs of integer bits quantizers (e.g., 2-level, 4-level, 8-level). More specifically, the quantizers which can adapt to the changing variance of its input by changing its step sizes has been investigated.

In general, two distinct classes of these quantizers exist; those that change their step size based on the transmitted value of the error signal and those that change their step size based on the variance of the unquantized error signal. The first form is known as "backward" quantizers since they look backward over previously quantized error samples to adjust their step size. The second form is labeled as "forward" quantizers because they look forward over the unquantized error sample to obtain their step size [13]. The backward quantizers need not send the quantizer step size to the receiver because the receiver can regenerate this value by looking at the transmitted sequence representing the quantized error waveform. This is not true of the forward quantizers. Here, since the step size is based on the value of the unquantized error signal, the receiver cannot regenerate it from the transmitted sequence. Thus, forward quantizers transmit the value of the quantizer to the receiver. Consequently, coders having forward quantizers require more bits than those having backward quantizers. For the SBAPC system, only the forward quantizers have been investigated. One of the reasons is that during the course of the channel error study, the SBAPC algorithm has been found to be extremely sensitive to errors. Since the backward quantizer is known to be more susceptible to errors as compared to the forward one, only the latter is considered [7].

In the SBAPC algorithm, the error waveform obtained after the pitch and predictor loops still has some sample-to-sample correlations. In particular, the error signal exhibits a large concentration of energy around the pitch pulses. If this pitch information can be finely quantized, the processed speech quality will be improved. One such example is the pitch-compensating quantizer [14] where two additional quantizer levels are especially designed to code large pitch pulses. Since the occurrence of pitch does not happen that often, a variable coding scheme has to be utilized to reduce the total bit rate and this complicates the entire coding process.

An alternative approach is to use a segmental quantization scheme which applies different quantization to various regions of the frame [15]. Partitioning the entire frame into a pre-determined number of sub-intervals, bit allocations can be computed adaptively according to the energies of these sub-regions in the same manner as dictated in Eq. (2-75). An 8-segment quantizer has been implemented in the SBAPC algorithm, and the results indicate that the segmental quantizer offers an advantage of 1 to 1.5 dB over that of the conventional quantizers. However, when bits needed to transmit the energies of the sub-levels are included in the algorithm (that is, less bits are utilized to quantize the residual signals), the improvement becomes minimal. In practice, the only situation that the segmental quantizer can offer any advantage is that it works pitch synchronously. In other words, more bits are utilized to quantize the larger pitch spikes especially during the onset of a pitch period. Unfortunately, there is no straightforward way to formulate such a pitch synchronous scheme owing to the fact that the pitch period is not always divisible by the number of sub-intervals. Consequently, the speech samples within a sub-interval cannot be guaranteed an integer making the transmission of a fixed number of bits per frame impossible.

In light of the fact that segmental quantizers do not offer any real advantage, conventional quantizers based on the statistics of the input signals are considered. The performance between the Gaussian [16] and the Laplacian [9] quantizers have been compared. As it turns out, the Gaussian quantizer consistently yields lower signal-to-quantization noise as compared to the Laplacian ones. This indicates that the SBAPC residual signals have a distribution which closely resembles a Laplacian one.

To further improve the SBAPC performance, quantizers have been derived from the actual distributions of the normalized low and high band error signals. Figures 2.5.3 and 2.5.4 depict one half of distribution of the low band, high band residual signal, respectively. From these figures, the actual distributions resemble the Laplacian more than the Gaussian ones. Also, the low band distribution seems to have a larger spread as compared to the high band one. Based on these distributions, the 1-bit, 2-bit, and 3-bit quantizers for the two bands have been obtained. When incorporated in the SBAPC algorithm, the new quantizers gain 0.25 dB S/Q. Also, informal listening tests indicate the new quantizers yield a "smoother" processed speech quality.

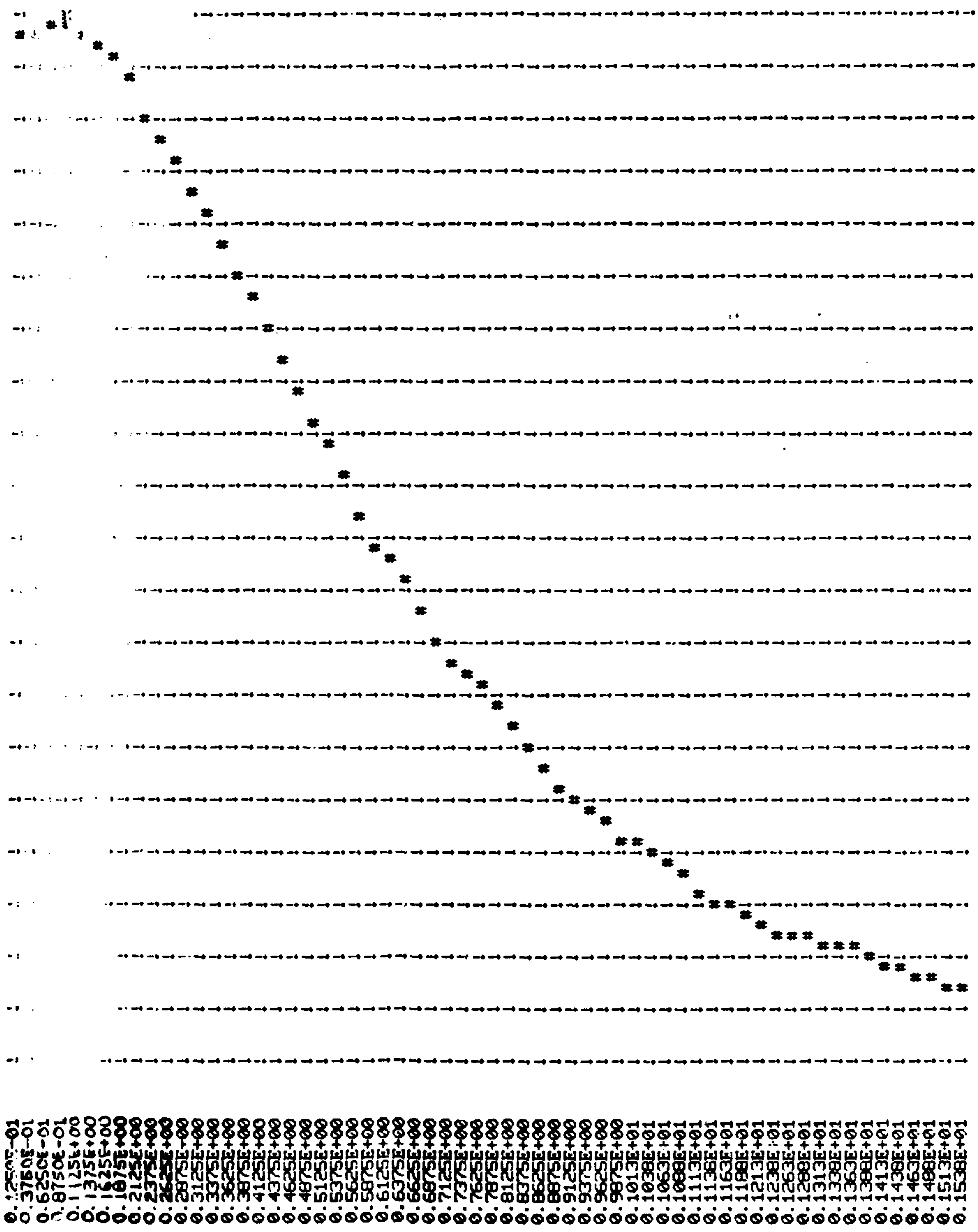


FIGURE 2.5.3: DISTRIBUTION OF THE LOW BAND RESIDUAL SIGNAL

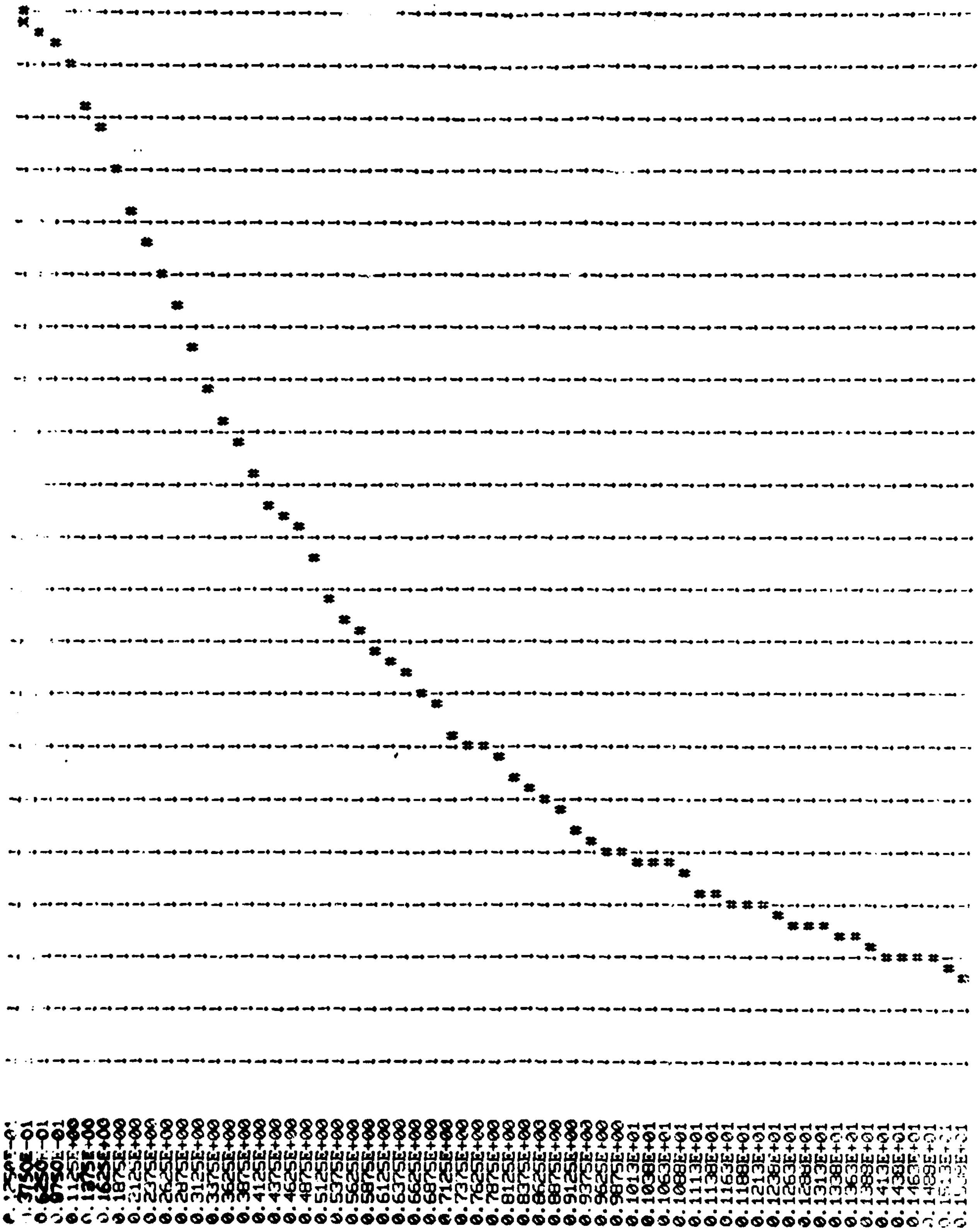


FIGURE 2.5.4: DISTRIBUTION OF THE HIGH BAND RESIDUAL SIGNAL

Section III

PERFORMANCE UNDER CHANNEL IMPAIRMENTS

3.1 The Effect of Background Noise on the SBAPC

It is well known that speech processing algorithms may produce high quality outputs with clean input speech materials, yet in many practical situations where the incoming signals are contaminated with acoustically coupled background noise, the quality of the speech processed through these algorithms can vary from slightly degraded to totally unintelligible. In this study, the effects of three types of background noises, namely, the office noise, the helicopter noise, and the P3C aircraft noise on the SPABC system have been investigated.

It has been shown in Section II that the SBAPC system yields high quality outputs at 16 Kb/s with uncorrupted spoken materials. Moreover, our result also indicates that the algorithm produces intelligible speech with noise coupled inputs even in high noise environments (S/N = - 6 dB). In the first part of this study, the original SBAPC algorithm is utilized to process the speech material supplied by DCA which contains standard sentences recorded in a low-level office environment (S/N = 90 dB). Informal listening tests show that the quality of the processed sentences is the same as that of clean inputs. As a matter of fact, quantization noise of the SBAPC tends to mask out the background office noise yielding smooth quality speech. This indicates that the SBAPC scheme can indeed function in an office environment and can be employed in the Executive Secure Voice Network (ESVN). The second part of the study deals with the utility of the SBAPC technique in a tactical surrounding simulated using helicopter noise and P3C aircraft noise. In both situations, the SBAPC yields highly intelligible speech even when the signal-to-noise ratios are as low as -6 dB. The periodic helicopter noise or the broad band P3C aircraft noise does not seem to have detrimental effects in the pitch extraction or the computation of predictor coefficients which render the processed speech unintelligible. However, the output material becomes extremely annoying to listen to especially for the high noise cases. So, for tactical situations, the algorithm has to be modified to include noise reduction techniques. Part 3 of this noise study deals with the design of a pre-processing scheme which is capable of suppressing the level of the background noise before inputting to the SBAPC algorithm.

3.1.1 Reduction of the Background Noise

In general, there are two types of methods that attempt to reduce the noise components from the corrupted speech signals. The first noise suppression scheme, generally known as the 2-microphone technique, employs a second microphone, which is far away from the speech source thus providing information about background noise alone [17]. It is then subtracted from the noisy input speech. This technique is effective for extremely high-noise environments (below 0 dB signal-to-noise ratio) and for non-stationary noise backgrounds (viz, means and correlation functions change rapidly in time). However, this scheme requires large amounts of computations (for example, the order of the noise cancelling filter is often greater than a thousand) which may be beyond the limits of real-time implementation. The second noise suppression technique [18] - [19], which uses a single microphone, estimates the frequency spectrum of the noise during non-speech activity. Then the noise spectrum is subtracted from that of the noisy speech. There are two approaches to suppress the background noise using the 1-microphone technique. The first one, proposed by Boll, attenuates the residual signal by -30 dB after subtraction of the estimated frequency components of the noise. The second procedure suggested by McAuley, reduces the residual signals depending on the frequency domain signal-to-noise ratio. Though the objective of both methods is to remove the stationary noise, the latter technique offers more versatility since the amount of noise reduced is controlled adaptively by a signal-to-noise ratio computed on a frame basis. In the next section, this technique and results of noise suppression by McAuley will be discussed.

3.1.2 McAuley's Noise Suppression Technique

Assuming that the noise n(t) has been added to the speech signal s(t), the computed input may be expressed as

$$x(k) = s(k) + n(k), \quad k=0, 1, \ldots, M-1 \tag{3-1}$$

where M is the number of speech samples in a frame period. Taking the DFT of eq. (3-1) yields

$$X(m) = S(m) + N(m), \quad m=0, 1, \ldots, M-1 \tag{3-2}$$

where X(m), S(m), and N(m) are the DFT's of x(k), s(k), and n(k), respectively. Assuming that the noise and speech signal are uncorrelated and they are sample functions of zero-mean Gaussian processes, then the variance of the X(m) may be expressed as

$$\sigma_X^2(m) = \sigma_S^2(m) + \sigma_N^2(m) \tag{3-3}$$

where $\sigma_S^2(m)$ and $\sigma_N^2(m)$ represent the variances of S(m) and N(m), respectively. Since X(m) is a complex Gaussian process with variance $\sigma_X^2(m)$, its real and imaginary parts are Gaussian with variances $\sigma_X^2(m)/2$. Therefore, the probability density function of X(m) may be expressed by the joint probability function:

$$p(X) = \frac{1}{\pi|\sigma_S^2 + \sigma_N^2|} \exp\left[-\frac{|X|^2}{\sigma_S^2 + \sigma_N^2}\right] \tag{3-4}$$

where the index m is omitted for simpler notation. The maximum likelihood estimate of σ_S^2 is obtained by differentiating p(X) with respect to σ_S^2 and setting the result to zero which yields

$$\hat{\sigma}_S^2 = |X|^2 - \sigma_N^2 \tag{3-5}$$

In order to reduce the distortion due to the phase, the input phase have to be retained, and the estimated spectral component of the signal may be expressed as

$$\hat{S}(m) = \hat{\sigma}_S(m) \frac{X(m)}{|X(m)|}$$

$$= \left[\frac{|X(m)|^2 - \sigma_N^2(m)}{|X(m)|^2}\right]^{1/2} X(m) \tag{3-6}$$

This is generally known as the method of spectral subtraction. Modifications of this algorithm have been studied extensively by several authors

[18] - [19]. The result of eq. (3-6) has been derived under the assumption that the speech and noise are independent Gaussian random processes.

The second approach is to assume that the speech can be characterized by a deterministic waveform with unknown amplitude and phase. In other words, only X(m) and N(m) as given in eq. (3-2) are random variables. Then the mean value of X(m) is given by

$$\bar{X}(m) = S(m) = A \exp(j\theta) \tag{3-7}$$

where A,θ is the amplitude, phase of the speech signal. Since N(m) is assumed to be zero-mean Gaussian, the probability density function of X(m) is written as

$$p(X|A,\theta) = \frac{1}{\pi\,\sigma_N^2}\ \exp\left[-\frac{|X|^2 - 2A\mathrm{Re}\{X\exp(-j\theta)\} + A^2}{\sigma_N^2}\right] \tag{3-8}$$

Assuming the phase θ is uniformly distributed over $[0,2\pi]$, then the probability density function of X given A may be expressed as

$$\begin{aligned} p(X|A) &= \int_0^{2\pi} p(X|A,\theta)\ \ p(\theta)\ d(\theta) \\ &= \frac{1}{\pi\sigma_N^2}\ \exp\left[-\frac{|X|^2 + A^2}{\sigma_N^2}\right] \cdot \frac{1}{2\pi}\int_0^{2\pi} \exp\left[\frac{2A\mathrm{Re}\{Xe^{-j\theta}\}}{\sigma_N^2}\right] d\theta \\ &= \frac{1}{\pi\sigma_N^2}\ \exp\left[-\frac{|X|^2 + A^2}{\sigma_N^2}\right] I_0\ (|2AX/\sigma_N^2|) \end{aligned} \tag{3-9}$$

where $I_0(.)$ is the zeroth order modified Bessel function of the first kind which is defined as

$$I_0(|X|) = \frac{1}{2\pi} \int_0^{2\pi} \exp\left[\mathrm{Re}(X\, e^{-j\theta})\right] d\theta \tag{3-10}$$

For large values of $|X| \geq 3$, $I_0(|X|)$ may be approximated by

$$I_0(|X|) \cong \frac{1}{\sqrt{2\pi|X|}} \exp|X| \tag{3-11}$$

In this case, the probability density function can be approximated as

$$p(X|A) \cong \frac{1}{\pi\sigma_N^2} \frac{1}{\sqrt{2\pi \frac{2A|X|}{\sigma_N^2}}} \exp\left[-\frac{|X|^2 - 2A|X| + A^2}{\sigma_N^2}\right] \tag{3-12}$$

The maximum likelihood estimate of the spectral amplitude may be obtained by differentiating eq. (3-12) with respect to A and by setting the result to zero which yields

$$\hat{A} = \frac{1}{2}\left[|X| + \sqrt{|X|^2 - \sigma_N^2}\right] \tag{3-13}$$

The maximum likelihood estimate of the spectral component without changing the phase may be expressed as

$$\hat{S}(m) = \hat{A} \frac{X(m)}{|X(m)|}$$

$$= \frac{1}{2}\left[1 + \sqrt{\frac{|X(m)|^2 - \sigma_N^2(m)}{|X(m)|^2}}\right] X(m) \tag{3-14}$$

In this case, the maximum likelihood estimate is the average of the received spectrum and the estimate spectrum obtained from the method of spectral subtraction. The relations between the received spectral amplitude A and the estimated amplitude $|X|$ given in eq. (3-13) are shown in Figure 3.1.1 for the cases of random input signal and deterministic input signal with unknown amplitude and phase. One advantage of the maximum likelihood algorithm is that the speech components at frequencies where the amplitude is small, is still preserved. In contrast, these components are removed completely in spectral subtraction technique which may degrade the quality of the speech or may decrease the intelligibility of speech.

However, the maximum likelihood algorithm does not adequately suppress the background noise in the absence of speech since the suppression rules are derived under the assumption that the speech signals are always present in the measured data. So a noise detector has to be developed in order to derive a better suppression rule that can be applied to reduce the noise component in the absence of speech signals. Instead of a fixed attenuation factor (-30 dB) in Boll's technique, an adaptive attenuation factor may be derived in this method. Modeling the speech activity as a hypothesis testing case, it can be represented as:

$$\begin{cases} H_0 : \text{ speech absent: } |X(m)| = |N(m)| \\ H_1 : \text{ speech present: } |X(m)| = |A\,e^{j\theta} + N(m)| \end{cases} \qquad (3\text{-}15)$$

Only the measured envelope is used in this measurement model since the measured phase provides no useful information in the suppression of noise. The spectral envelope estimate, A, derived from the minimization of the

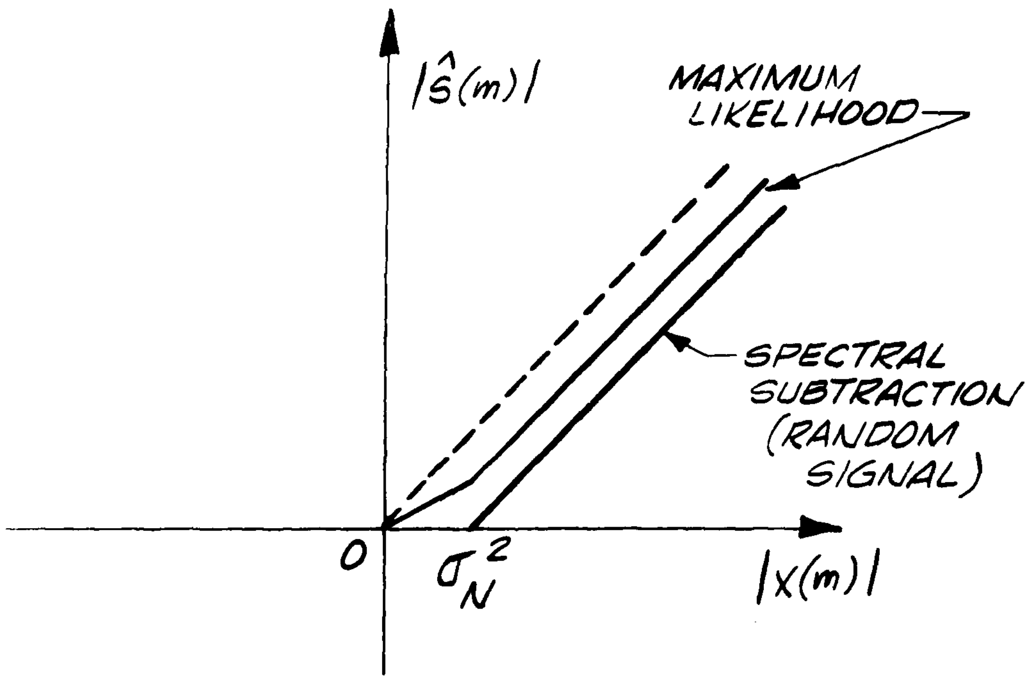


FIGURE 3.1.1 TRANSFER CHARACTERISTICS OF THE NOISE SUPPRESSION DEVICE

mean-squared error $E[(A - \hat{A})^2]$ is given as $E[A \mid |X|]$. This conditional mean can be expressed as

$$\hat{A} = E[A \mid |X|]$$

$$= E[A \mid |X|, H_1] P[H_1 \mid |X|] + E[A \mid |X|, H_0] P[H_0 \mid |X|] \tag{3-16}$$

where $P[H_k \mid |X|]$ is the probability that the speech is classified as in state H_k. The last term of eq. (3-16) becomes zero due to the fact that the average value of A when the speech is not present should be equal to zero. Then the estimate of the envelope A is given by

$$\hat{A} = E[A \mid |X|, H_1] \; P(H_1 \mid |X|) \tag{3-17}$$

When speech is present, the expectrum spectrum $E[A \mid |X|, H_1]$ represents the minimum variance estimate of A and it can be substituted with the maximum likelihood estimate given in eq. (3-13) which results:

$$\hat{A} \simeq \frac{1}{2}\left[|X| + \sqrt{|X|^2 - \sigma_N^2} \right] P\left[H_1 \mid |X| \right] \tag{3-18}$$

where $P[H_1 \mid |X|]$ may be expressed as:

$$P[H_1 \mid |X|] = \frac{P[|X| \mid H_1] \; P[H_1]}{P[|X| \mid H_1] \; P[H_1] + P[|X| \mid H_0] \; P[H_0]} \tag{3-19}$$

Under hypothesis H_0, the received signal envelope consists of noise term only. Since the noise is a complex Gaussian process with a zero mean and a variance σ_N^2, the envelope will have the Rayleigh probability density function and can be written as

$$p(|X| \big| H_0) = \frac{2|X|}{\sigma_N^2} \exp\left[- \frac{|X|^2}{\sigma_N^2} \right] \tag{3-20}$$

Under hypothesis H_1, the probability density function of the received envelope will have the Rician density function and may be expressed as

$$p(|X| \big| H_1) = \frac{2\ X}{\sigma_N^2} \exp\left[- \frac{|X|^2 + A^2}{\sigma_N^2} \right] I_0 \left[\frac{2A|X|}{\sigma_N^2} \right] \tag{3-21}$$

Assuming that a priori probabilities of the hypothesis $P[H_0]$, $P[H_1]$ are equal and defining the a priori signal-to-noise ratio to be

$$\xi = \frac{A^2}{\sigma_N^2} \tag{3-22}$$

Equation (3-19) can be rewritten as

$$p(H_1 \big| |X|) = \frac{\exp\ (-\xi)\ I_0 \left[2\sqrt{\xi\ (|X|^2/\sigma_N^2)} \right]}{1 + \exp\ (-\xi)\ I_0 \left[2\sqrt{\xi\,(|X|^2/\sigma_N^2)} \right]} \tag{3-23}$$

It is this a-posteriori-probability that contributes the "soft-decision" aspect to the maximum likelihood envelope estimator as compared to the "hard decision" of eq. (3-13) for which the speech plus noise is either passed or is blocked depending on the decision of the hypothesis. Defining a posteriori signal-to-noise ratio (i.e., measured signal-to-noise ratio SNR) as

$$SNR = |X|^2/\sigma_N^2 \tag{3-24}$$

and appending the measured input phase, the estimated spectral component may be expressed as

$$\hat{S}(m) = |\hat{A}(m)| \frac{X(m)}{|X(m)|}$$

$$= \frac{1}{2}\left[1 + \frac{\sqrt{|X(m)|^2 - \sigma_N^2}}{|X(m)|}\right] \cdot X(m) \cdot p\left[H_1 \middle| |X(m)|\right]$$

$$= \frac{1}{2}\left[1 + \sqrt{\frac{SNR-1}{SNR}}\right] \frac{\exp(-\xi)\, I_0(2\sqrt{\xi \cdot SNR})}{1 + \exp(-\xi)\, I_0(2\sqrt{\xi \cdot SNR})} X(m) \tag{3-25}$$

The channel gains, given as the multiplication of the a-posteriori probability for the speech state $P[H_1 | |X|]$ by the maximum likelihood envelope estimate of eq. (3-13), are plotted in Figure 3.1.2 as a function of a-posteriori signal-to-noise ratio (SNR) for various values of a priori signal-to-noise ratio ξ. The two-state soft-decision maximum likelihood algorithm applies more suppression when the measured SNR is low and this case "most likely" corresponds to the noise state. On the other hand, little attenuation is applied when the SNR is large. This is a desirable property of the noise suppression device, since the state of large SNR "most likely" means that speech is present, in which little attenuation is desired. As ξ increases, the channel gain curves become sharper which indicate that the speech state (H_0 or H_1) can be distinguished easier for large ξ. In the limit, the output may be totally suppressed or passed depending on the value of a measured a posteriori signal-to-noise ratio. This particular case leads to the similar performance of Boll's noise suppression algorithm whose attenuation factor depends solely on the decision of

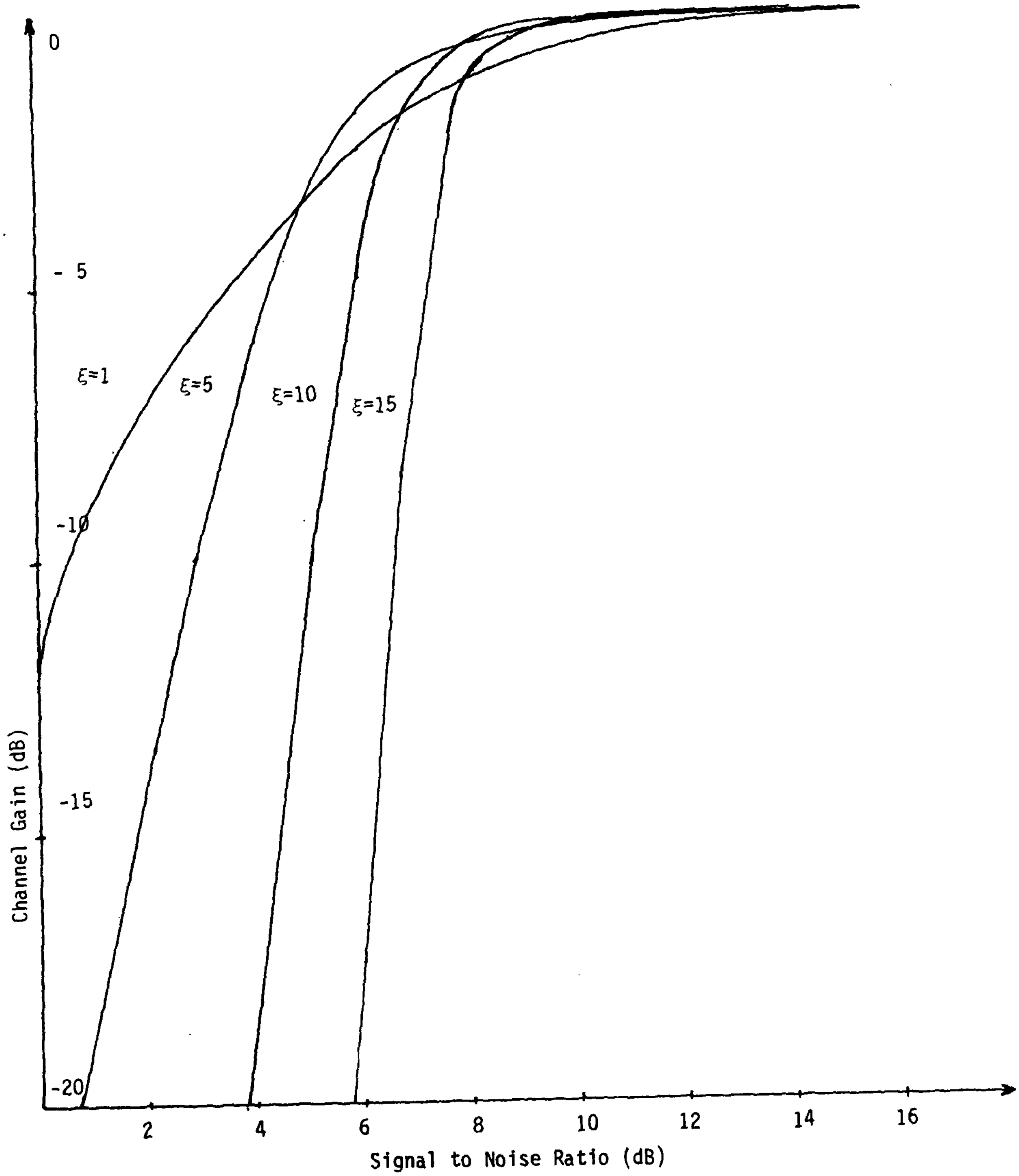


FIGURE 3.1.2: THE CHARACTERISTIC FUNCTION OF THE MAXIMUM LIKELIHOOD NOISE SUPPRESSION

speech state. With the selection of ξ, the McAuley technique provides the versatility of noise suppression in an adaptive fashion. It is, therefore, convenient to refer ξ as the "suppression factor" which is chosen according to the background noise level. Once ξ is chosen, the a posteriori signal-to-noise ratio must be measured in order to calculate the channel gain as shown in eq. (3-25).

The block diagram of the noise suppression technique is shown in Figure 3.1.3. In this scheme, the energy of the input signal is computed and fed to the noise detector which decides the speech activity, i.e., speech present or speech not present (noise) using the detection algorithm as discussed in Appendix B. Concurrently, the spectra of the input signal X(k) are calculated via the FFT (fast Fourier transform) technique. The resulting spectral components are directed to the spectral mean adjustment device when the speech is not present. The multiplication gain factors are calculated for each spectral component from the input spectra and the average noise spectra. These gain factors are then multiplied with the input spectra, and the estimated signal is obtained via inverse FFT.

Mathematically, let the average noise power at the mth frame and nth channel be:

$$\lambda(m) = \lambda_n(m-1) + \alpha \left[|X(m)|_n^2 - \lambda_n(m-1) \right] \tag{3-26}$$

where $\lambda(m) = \sigma_N^2(m)$ is used for notational convenience and $|X(m)|_n^2$ represents the measured noise power spectrum at the nth channel of the mth frame. Then, the average noise power is updated after each frame using the time constant about 1 sec., i.e.,

AD-A092 010 GTE PRODUCTS CORP NEEDHAM HEIGHTS MA COMMUNICATION S--ETC F/G 9/4
SPEECH ALGORITHM OPTIMIZATION AT 16 KBPS.(U)
SEP 80 R S CHEUNG, S Y KWON, A J GOLDBERG DCA100-79-C-0038

UNCLASSIFIED NL

2 of 3
AD A
092010

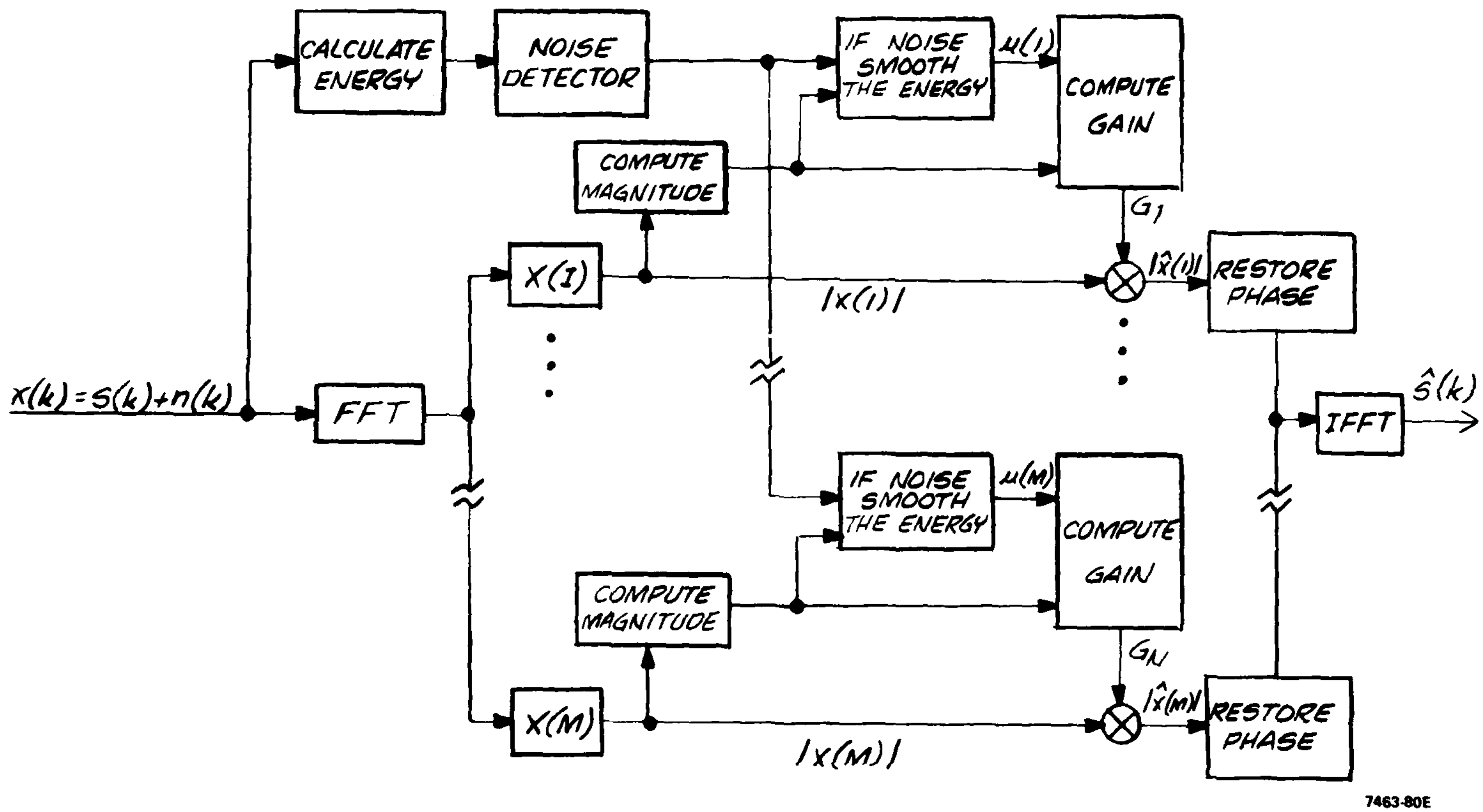


FIGURE 3.1.3 BLOCK DIAGRAM OF McAULEY'S NOISE SUPPRESSION ALGORITHM

$$\alpha = \exp|-22.5/T| \tag{3-27}$$

where 22.5 represents a frame period in msec and T is the time constant in m sec. One of the disadvantages of this scheme is the relatively long adaptation time required to determine the detection threshold and then additional training period may be required to learn the channel noise statistics. Let the gain factor of the spectral subtraction be

$$g_n(m) = \frac{|X(m)|_n^2 - \lambda_n(m-1)}{|X(m)|_n^2} \tag{3-28}$$

Then the channel gain can be expressed from eq. (3-25) as

$$G_n(m) = \frac{|\hat{S}_n(m)|}{|X(m)|_n}$$

$$= \frac{1}{2}\left(1 + \sqrt{g_n(m)}\right) \frac{\exp(-\xi)\, I_0\left(2\sqrt{\dfrac{\xi}{1 - g_n(m)}}\right)}{1 + \exp(-\xi)\, I_0\left(2\sqrt{\dfrac{\xi}{1 - g_n(m)}}\right)} \tag{3-29}$$

The advantage of using $g_n(m)$ as an independent variable is that the value of $g_n(m)$ is less than one, which facilitates the computation of the channel gain using a simple table look-up program. Fifteen tables corresponding to values $\xi = 1, 2, \ldots, 15$ have been tabulated in the noise suppression algorithm, with each table consisting of 50 values of suppression rule computed for equal increment of $g_n(m)$ from 0 to 1.

The McAuley algorithm has been applied to the processing of speech signals added with various types of background noise with the 16 Kb/s SBAPC system. The output of the noise suppression device or the synthesizer speech is noted to have amplitude fadings when large ξ is used in the high noise environment (S/N < 0 dB). This is a very objectionable degradation of speech. In order to maintain a constant amplitude output even for

large ξ, a simple automatic gain control (AGC) routine has to be incorporated with the noise suppression algorithm. If $G_A(m)$ denotes the average channel gain at the mth frame, i.e.,

$$G_A(m) = \frac{1}{M} \sum_{n=0}^{M-1} G_n(m) \tag{3-30}$$

where M is the channel number in frequency domain. This average gain factor may indicate approximately the amounts of the spectral suppression. A small number of $G_A(m)$ may correspond to a large amount of power attenuation. In this case, the output may need a large amplification to avoid the fading of amplitudes. Furthermore, to maintain a constant gain throughout all the frames, a smoothing algorithm is utilized and the overall gain becomes:

$$G(m) = \frac{\frac{1}{128} \sum_{\ell=0}^{129} G_A(m-\ell)}{\frac{1}{4} \sum_{\ell=0}^{3} G_A(m-\ell)} \tag{3-31}$$

In our simulations, the effects of amplitude fading have been reduced even in the high noise case (S/N = -6 dB) where the gain factor G(m) is utilized to adjust the output of the noise suppression device. The noise suppression algorithm developed in this project can be used as a preprocessor to any speech signal processing algorithm, and the overall system can be further optimized with respect to the noise suppression/speech distortion tradeoff analysis by choosing an appropriate suppression factor ξ.

3.2 The Effect of Random Channel Errors on the SBAPC

The SBAPC system, as discussed in Section II, produces a high quality synthesized speech at the data rate of 16 Kbps. However, error-free transmissions are not always possible in many practical systems, since the transmitted signals may be corrupted by noises in the channel which may or may not vary with time. Under these circumstances, the performance of the SBAPC changes greatly with the rates and also with the positions of the channel errors within the frame.

In this study, the effect of random channel errors at rates ranging from 0 - 10^{-2} is investigated. The configuration of the SBAPC system tested ($\approx$15 Kbps) is shown in Figure 3.2.1 which includes the 32-tap QMF, 1st order pitch loop on the low band, 4th order APC on both bands, noise shaping on both bands, and adaptive allocation of 288 bits on quantizing the subband signals. The signal-to-noise ratio plot versus bit error rates (BER) is shown in Figure 3.2.2. The result indicates that the SBAPC system is extremely sensitive to channel errors. At 10^{-4} BER, the algorithm's performance is virtually unchanged as compared to the no error case. However, the degradation becomes noticeable at 5 x 10^{-4} BER, and the signal-to-noise ratio drops by 5 dB at 10^{-3}. At 10^{-2} BER, the system is useless since the output speech becomes unintelligible. As expected, errors occurred on the side information bits which include pitch, PARCOR coefficients, energy, etc., have a more detrimental effect as compared to those occurred on the quantized residual signal bit stream. This suggests that in order to make the SBAPC system useful in a noisy channel, protections of the transmitted bits are vital, particularly for the side information bits. The following section describes the utility of forward error correcting codes in reducing the effect of channel errors.

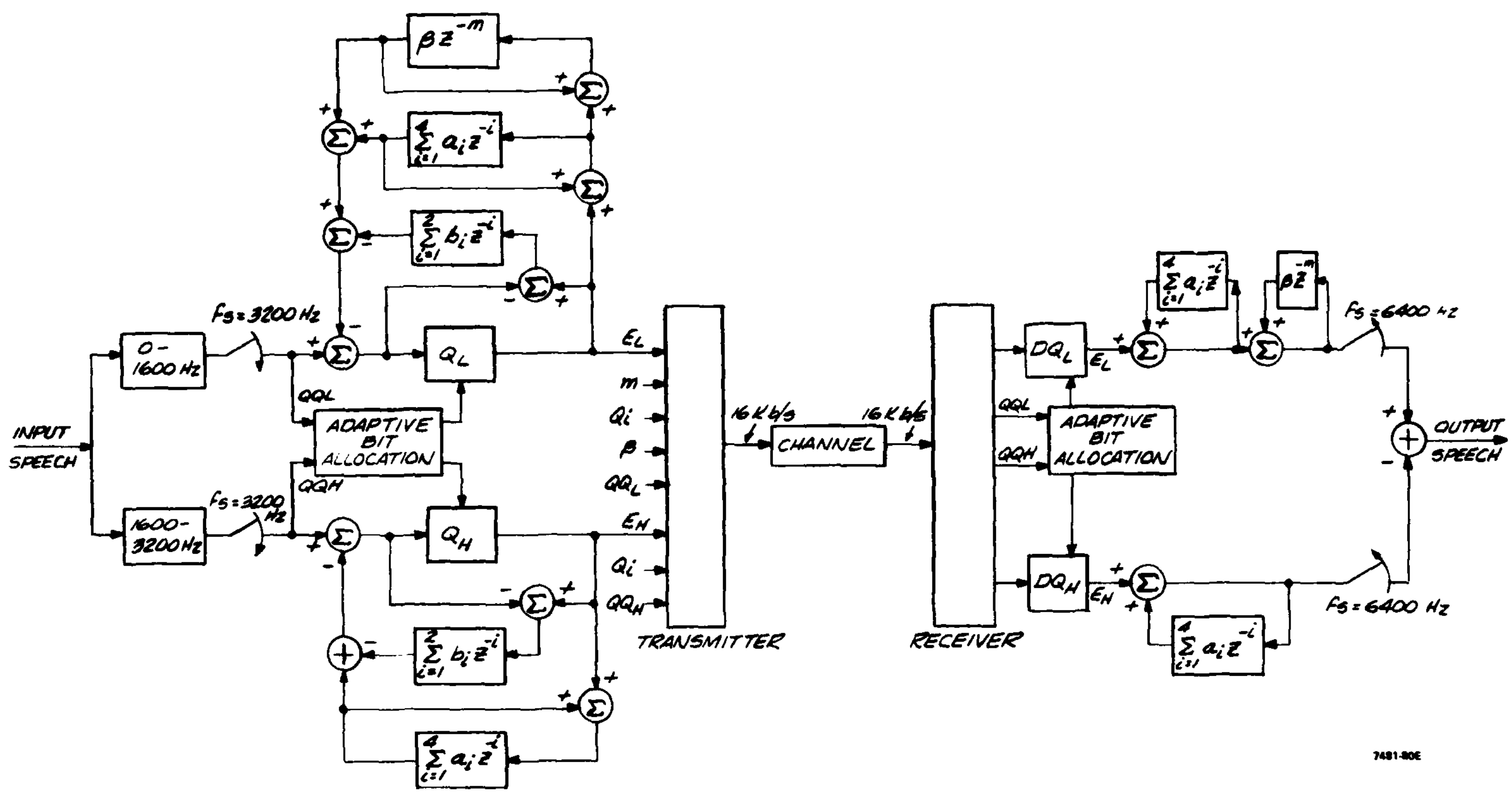


FIGURE 3.2.1 THE 16 KBPS SPLIT-BAND APC SYSTEM

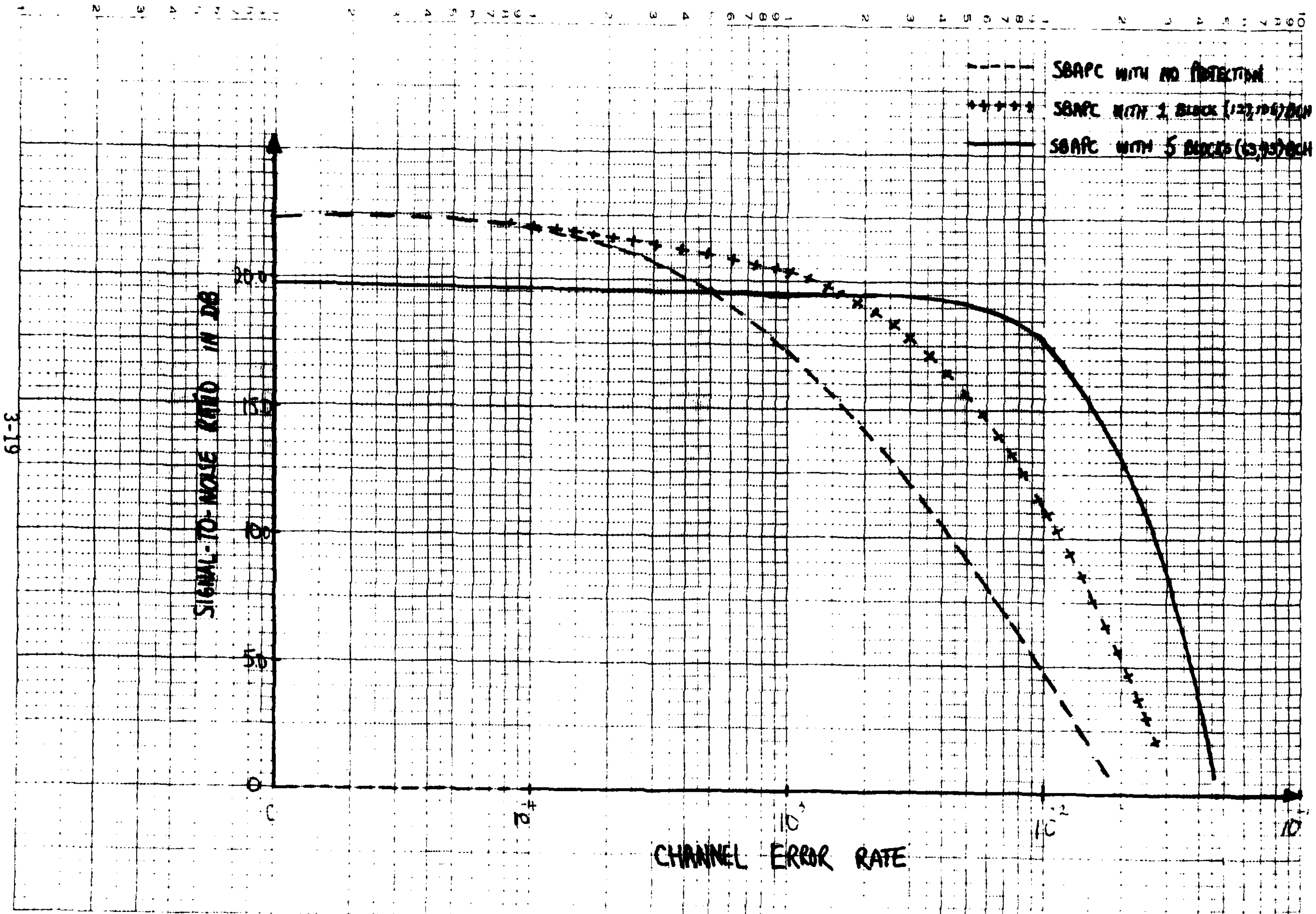


FIGURE 3.2.2: THE EFFECT OF CHANNEL ERROR RATES ON 3 SBAPC SYSTEMS

3.2.1 Application of BCH Codes

The method of correcting errors may be chosen depending on the application circumstances, i.e., data rate, channel error rate, complexity and cost, etc. Since the SBAPC coder is designed for real-time implementations, it is desirable to have error correcting codes that require the least amount of time delay in correcting channel errors. Block codes of short length may be well suited to the real-time implementation of the SBAPC algorithm since no additional time delay is required to process the error correcting procedure if the length of the block code is less than the number of bits received in a frame period. There are many types of block codes that can be utilized for different channel characteristics.

Practical communication channels corrupt signals in many ways, such as the additive Gaussian noise and/or impulsive noise that produce random and burst errors, respectively. In other situations, the characteristics of the channel may vary in time (fading HF channels) or may be random since it represents a sample function of an ensemble of channels with widely different characteristics (switched telephone network). Hence, it is non-trivial to construct a coding scheme that adapts to various types of channels. Since additive Gaussian noise is the main source of noise in many practical communication channels, only forward error correcting codes that are capable of correcting random errors are considered.

The Base-Chaudhuri-Hocquenhem (BCH) code, which is a remarkable generalization of Hamming codes, has been known to be the most powerful multiple random-error correcting code. Also, the decoding algorithm can be implemented with a reasonable amount of complexity
A more fundamental description of the BCH codes and their encoding, decoding algorithms are given in Appendix C. This appendix has shown that

with the block length of $n = 2^m - 1$ and mt parity bits it is possible to correct any t or less errors using a (n, k) BCH code where k is the number of information bits. The proper choice of m, n, t for BCH codes may depend on the channel error rate, data rate, and the system's specifications. The information rate of the (n, k) BCH code is given as:

$$R = k/n \tag{3-32}$$

The performance of random-error correcting BCH codes may be expressed in terms of error-probability. Let P(m, n) be the probability of m errors occurring in an n-bit block and β_m be the probability of decoding an error pattern of weight m correctly, then the probability of decoding received code word erroneously may be expressed as

$$\begin{aligned} P_e &= 1 - \sum_{m=0}^{n} \beta_m P(m, n) \\ &= \sum_{m=0}^{n} \alpha_m P(m, n) \end{aligned} \tag{3-33}$$

where $\alpha_m = 1 - \beta_m$ denotes the probability of erroneously decoding an error pattern of weight m. The parameter α_m is a function of the code and decoding algorithm. If a t error-correcting BCH code is employed and it is decoded using the Peterson decoding algorithm shown in Appendix D, the parameter α_m may be expressed as

$$\begin{aligned} \alpha_m &= 0 \qquad 0 \leq m \leq t \\ &= 1 \qquad t < m \leq n \end{aligned} \tag{3-34}$$

and the probability of erroneously decoding the code word may be reduced from eq. (3-33) as

$$P_e = \sum_{m=t+1}^{n} p(m, n) \tag{3-35}$$

If the bit errors occur independently at random with probability e, then the probability p(m, n) can be expressed as

$$p(m, n) = \sum_{m}^{n} e^{m}(1-e)^{n-m} \tag{3-36}$$

where the probability p(m, n) is simply the binomial distribution and P_e in eq. (3-35) is equal to the tail of the distribution.

3.2.2 Error Protection via the (127,106) BCH code

This technique employs one block of three-error correcting (127,106) BCH code which protects 106 information bits with 21 parity ones. For the 16 Kb/s SBAPC system, all 50 side information bits, together with 56 sign bits of the residual signals are encoded. A sync bit and 232 error signal bits are left unprotected. Though this coding scheme is efficient (i.e., only 5.8% of the total bits are used for error protection), its success hinges largely on the assumption that the SBAPC system is tolerant to random errors occurred on the residual signal bit stream. The S/Q plot versus BER for the SBAPC system with the (127,106) BCH Code is depicted by the graph (+) in Figure 3.2.2. As illustrated in the figure, the protected SBAPC system consistently out-performs the original unmodified one in high error cases. For the system with the (127,106) BCH code, the S/Q remains relatively unchanged for BER from 0 - 10^{-3}. Unfortunately, its performance starts to degrade at 2×10^{-3} BER, and poor quality, though intelligible, speech is obtained at 10^{-2} BER. This result reveals two important findings: 1) protection of only side information is not even adequate in maintaining the SBAPC performance in relatively low-error environments; 2) the utilization of long block BCH codes is not the best strategy for high-error situations (e.g., BER = 10^{-2}). The first finding can be attributed to the presence of the pitch loop in the SBAPC algorithm which propagates the residual signal errors to successive frames. One solution to the above situation is to apply error protection to residual signal bits as well as to side information bits at the expense of higher transmission rates The second finding results from the fact that the occurrence of channel errors is more likely in a longer data block. If this error count exceeds the correcting capability of the BCH code ($t = 3$ for the (127,106) BCH code), the code renders no utility. To illustrate this, the probability of more than 3 error occurrence in the block of 127 bits at 10^{-2} channel error rate is computed using eq. (3-36) as follows:

$$
\begin{aligned}
P_e &= \sum_{m=4}^{127} p(m, 127) \\
&= 1 - p(0,127) - p(1,127) - p(2,127) - p(3,127) \qquad (3\text{-}37) \\
&= 0.0393
\end{aligned}
$$

On the average, there will be 4 blocks out of every 100 that will have more than 3 errors, and they will not be corrected. Furthermore, the presence of the pitch loop in the SBAPC algorithm compounds the effects by propagating the errors through several frames. One solution to overcome the above deficiency is the utilization of several blocks of short BCH codes e.g., (63,45). To illustrate this, the probability of more than 3-error occurrence in the block of 63 bits at 10^{-2} channel error rate is computed using eq. (3-36) as follows:

$$P_e = \sum_{m=4}^{63} p(m, 63)$$

$$= 1 - p(0,63) - p(1,63) - p(2,63) - p(3,63) \qquad (3\text{-}38)$$

$$= 0.003725$$

This indicates that on the average, 4 blocks out of every 1000 will have more than 3 errors, making this code an order of magnitude more resistant to channel errors than the (127,106) BCH code.

3.2.3 Error Protection via Five Blocks of (63,45) BCH Codes

As discussed in Section 3.2.2, the incorporation of 1 block of (127,106) BCH code to protect the 50 side information bits and 50 residual signal bits does extend the utility of the SBAPC system from 10^{-3} to 5×10^{-3} channel error rate. However, at 10^{-2}, its performance is still considered unacceptable. One alternative is to use shorter length BCH codes to maintain their error correcting capability in high error environments. Also, the employment of multiple blocks of these short BCH codes to protect more residual signal bits will further enhance the SBAPC system's robustness to channel errors. The following describes a forward error-correcting procedure that employs 5 blocks of (63,45) BCH codes, and it extends the utility of the SBAPC algorithm to 10^{-2} BER.

Since 5 blocks of (63,45) BCH codes require 90 parity bits, the SBAPC algorithm has to be modified slightly in order to maintain a transmission rate of 16 Kbps (360 bits per frame at 44.44 frames/sec.). In this new configuration, 50 bits/frame are for side information quantization, 216 bits/frame are for encoding residual signals, 90 bits are needed for parity checks, and the remaining 4 bits are for synchronization purposes. As for the error protection, 90 parity bits are used to defend 50 side information bits and 175 residual signal bits. The signal-to-quantization noise plot versus bit error rates is shown in Figure 3.2.2. Compared to the SBAPC system with no protection or that with 1 block of (127,106) code, the multiple-block encoding scheme yields slightly inferior S/Q for low channel error rates (BER $\leq 5 \times 10^{-3}$). This result is not surprising since a higher percentage (25%) of the available bits are spent on parity checking rather than on quantizing the residual signals. Consequently, the subband error signals are represented less precisely yielding lower S/Q. However, informal listening tests reveal no or little audible differences between the processed sentences obtained through the SBAPC system with 5 blocks of (63,45) BCH codes and the SBPAC with 1 block of (127,106) code in the error-free case. This may be explained by the fact that the S/Q for the SBAPC with multiple-block error coding is already high ($\simeq$ 19.6 dB). An additional 2.5 dB increase obtained from the SBAPC with 1 block error coding is not sufficient to perceptually improve voice quality.

For channels with high error rates, the multiple block coding scheme outperforms the one block system by as much as 7 dB, and it yields high quality speech even at 10^{-2} BER. So, employing the 5-block (63,45) BCH coding scheme, the performance of the 16 Kb/s SBAPC system can be made robust to channel error rates as high as 10^{-2}.

3.3 Tandem Performance with 2.4 Kb/s LPC-10

In daily communications, users of the 16 Kb/s wideband terminals may have to converse with those of the 2.4 Kb/s narrowband ones. Though these terminals may individually produce satisfactory outputs, the overall speech quality when they are in connection or in tandem is sometimes degraded. This type of distortion is exemplified by the "buzzy" speech quality obtained when the 16 Kb/s Tenley terminal, which employs the Continuously Variable Slope Deltamodulator (CVSD), is connected with the 2.4 Kb/s STU-2 terminal which encodes speech using the LPC-10. The degradation may be partly attributed to the algorithms which have been optimized only for clean input speech. Moreover, it may also be due to the interactions of distortions introduced by the first speech encoding scheme with that of the subsequent terminals. So, in order for the 16 Kb/s SBAPC algorithms to provide greater utility, good tandem performance with LPC is a definite requirement.

APC schemes are known to tandem well with LPC [20]. Since the SBAPC algorithm is a modified form of APC, it also exhibits no undesirable distortions when connected with LPC. In particular, when the peaky LPC synthesized waveform is fed into the SBAPC, a smooth but slightly low-passed quality speech results. When compared to the processed material obtained from the LPC/CVSD tandem, the speech quality of the LPC/SBAPC tandem is less muffled since the SBAPC algorithm does not produce slope overloading. On the other hand, when the SBAPC synthesized speech is fed into LPC, outputs similar to that derived from LPC alone are obtained. The SBAPC/LPC is much more pleasant to listen to than the "buzzy" quality of the CVSD/LPC tandem.

SECTION IV

FORTRAN SIMULATIONS

4.1 FORTRAN Simulations of the SBAPC System

The SBAPC algorithm, as depicted in Figure 3.2.1, has been simulated on a PDP 11/70 computer using FORTRAN IV-PLUS. The flow diagram of the program is summarized in Figure 4.1.1. Input speech, previously digitized and stored on disks or magnetic tapes, is processed by the SBAPC program and the output material is also written back on disks or tapes. Operations of the transmitter include noise suppression, quadrature mirror filtering with down-sampling, computation of low band predictor coefficients, computations of high band predictor coefficients, adaptive bit allocations, low band APC analyzer with noise shaping, high band APC analyzer with noise shaping, serialization, and BCH encoder. At the receiver, transmissions of wrong binary bits are corrected via a BCH decoder. After deserialization and dequantization, the received residual signals and APC parameters generate estimates of the low band and the high band waveforms via their corresponding APC synthesizers. The output speech is then obtained by up-sampling the two subband signals together with quadrature mirror filtering.

At the transmitter, the speech samples are brought in (144 + 18) at a time, but only 144 samples correspond to the new frame. The other 18 samples belong to the previous frame and they are employed for smoothing frame boundaries. Then the new data are processed through the noise suppression routine whose flowchart is shown in Figure 4.1.2. Initially, the energy and the spectrum of the input signal are computed. According to its energy, the decision on whether the frame is silence, noise only or speech with noise is made with the help of the modified Robert's algorithm in

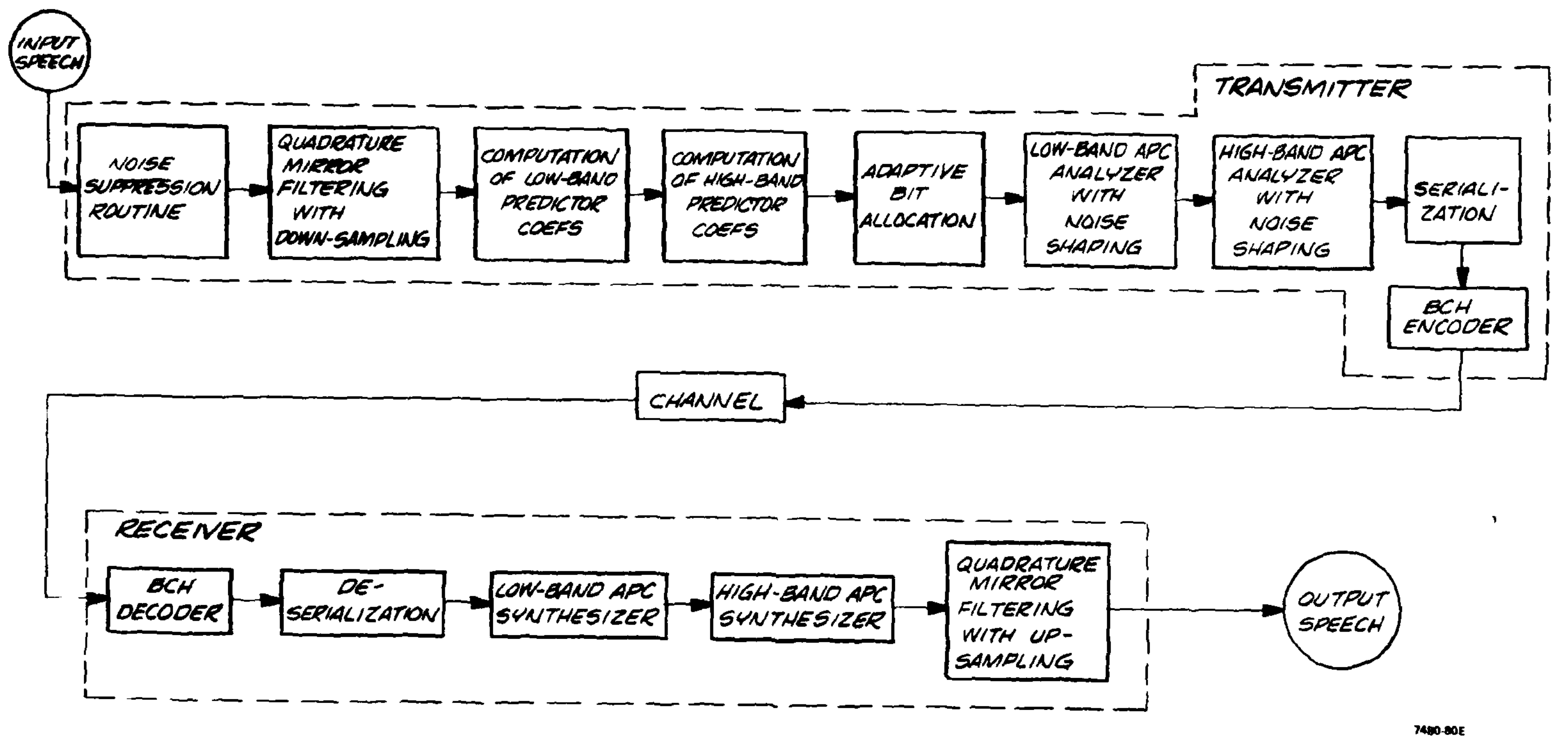


FIGURE 4.1.1 FLOW DIAGRAM OF THE 16 KBPS SBAPC FORTRAN PROGRAM

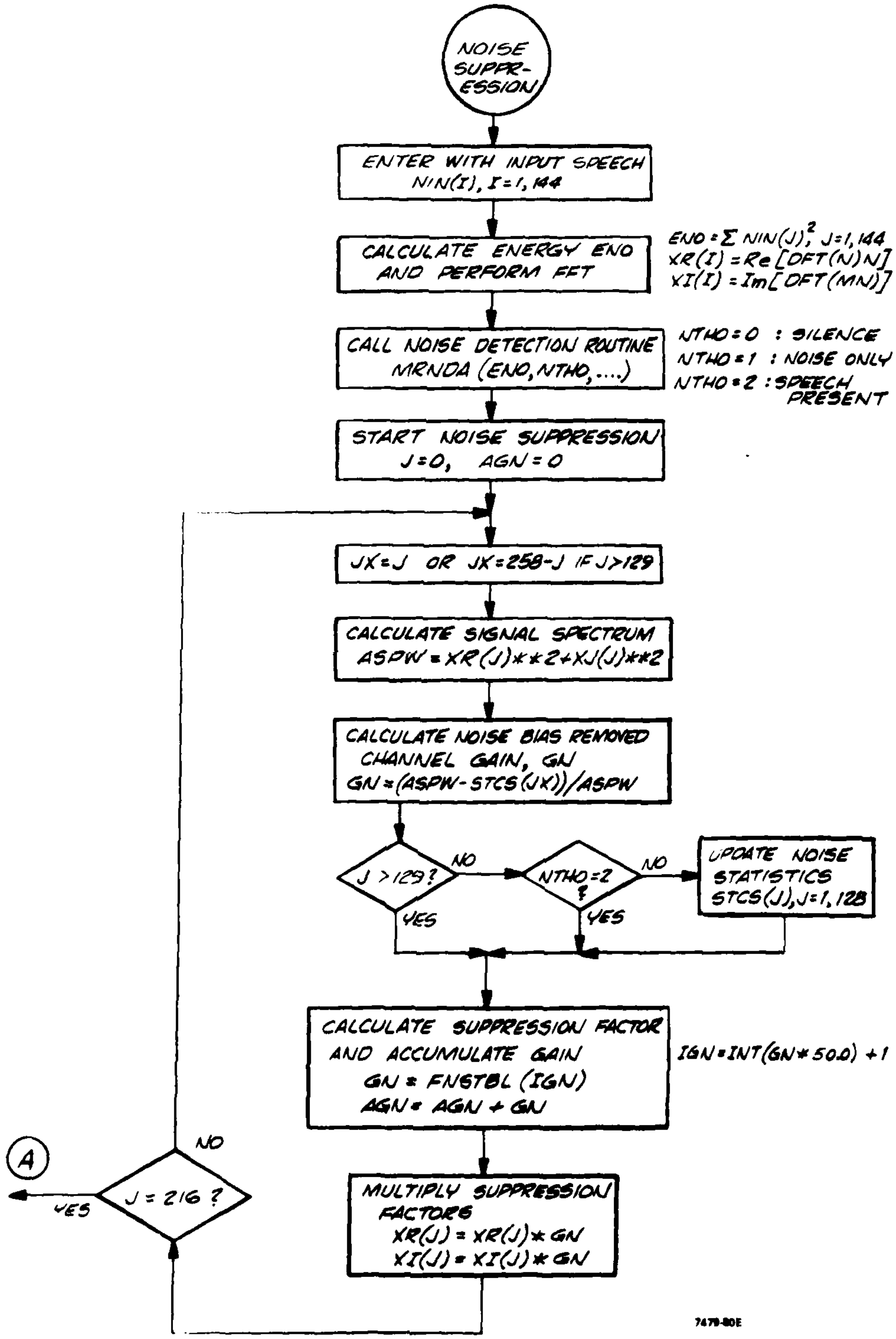


FIGURE 4.1.2 FLOW CHART OF NOISE SUPPRESSION ROUTINE

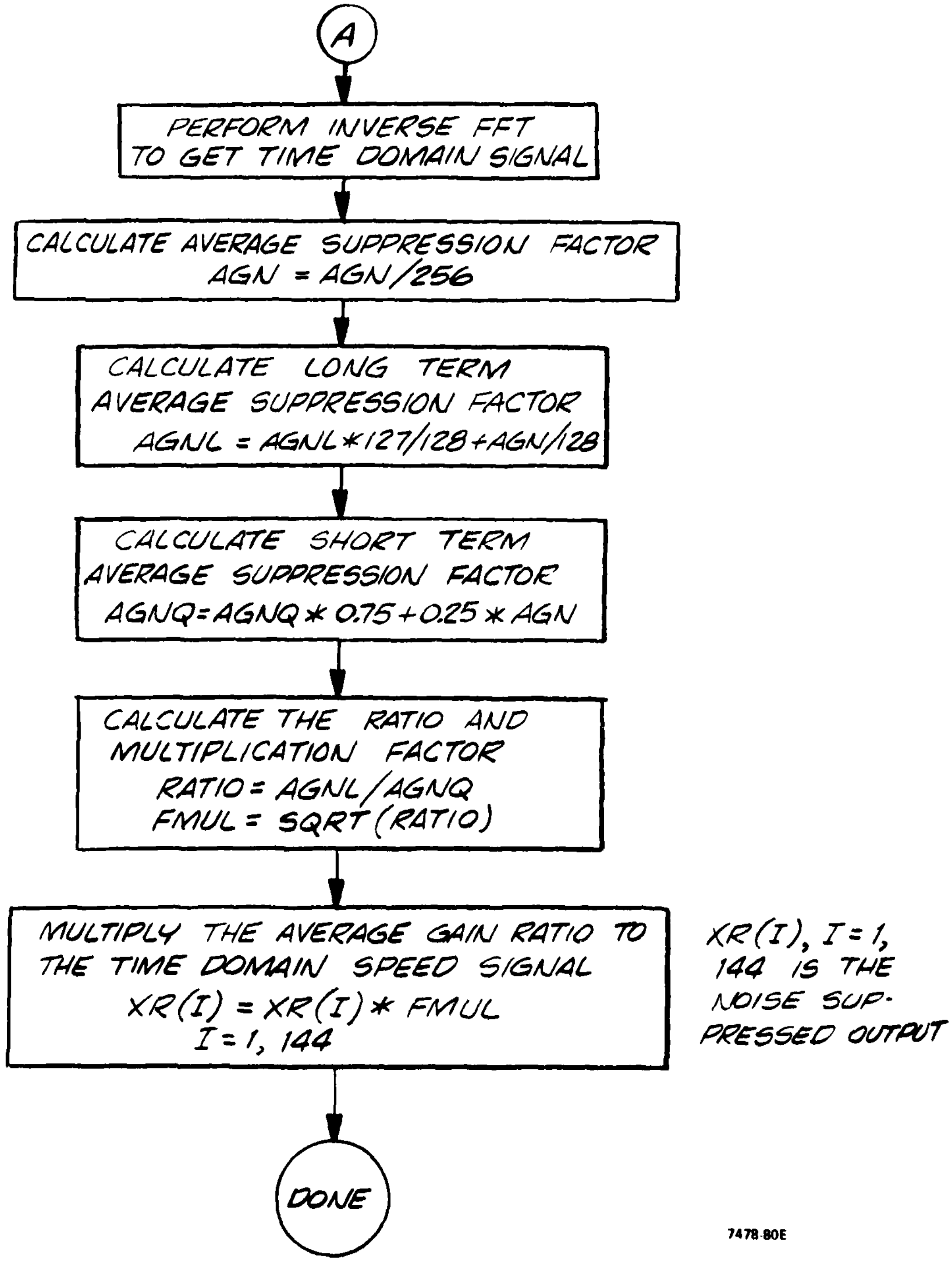


FIGURE 4.1.2 FLOW CHART OF THE NOISE SUPPRESSION ROUTINE (Cont.)

Appendix B. Based on this decision, the noise statistics are updated, and the noisy components are suppressed from the incoming signal using the McAuley algorithm as discussed in Section 3.2. Then the noise-reduced speech is fed into the SBAPC coder.

The first operation in the coder is to split the frequency band of the incoming signal into two subbands via quadrature mirror filtering in the manner as depicted in Figure 4.1.3. The waveforms of the low and high bands after down sampling are encoded using APC. The computation of the four low band APC coefficients are performed as shown in Figure 4.1.4 which includes the pitch extraction via the autocorrelation technique, the calculation of pitch gain, and the determination of PARCOR coefficients from the reduced waveform using the Levinson recursion. Similarly, the computation of the four high band APC coefficients are done as depicted in Figure 4.1.5. In contrast to the low band case, no pitch loop is necessary in the high band. After computing the filter coefficients, the prediction residual energies (QQL, QQH) for the two bands are utilized for quantizer bit allocation. The adaptive rule is detailed in Figure 4.1.6. With the definition of quantizer bit assignments, the APC residual signals for both bands are generated and quantized as illustrated in Figure 4.1.7. Makhoul's second order all-zero filter is also incorporated for shaping the quantizing noise. After serializing the quantized parameters into a bit stream, 5 blocks of (63,45) BCH codes (90 parity bits) are employed to encode 50 side information and 175 error signal bits as shown in Figure 4.1.8.

At the receiver, the reverse of the transmitter operations are performed. After correcting the transmission errors via the BCH decoder whose flowcharts are included in Figures C.1 and C.2, the bit stream is

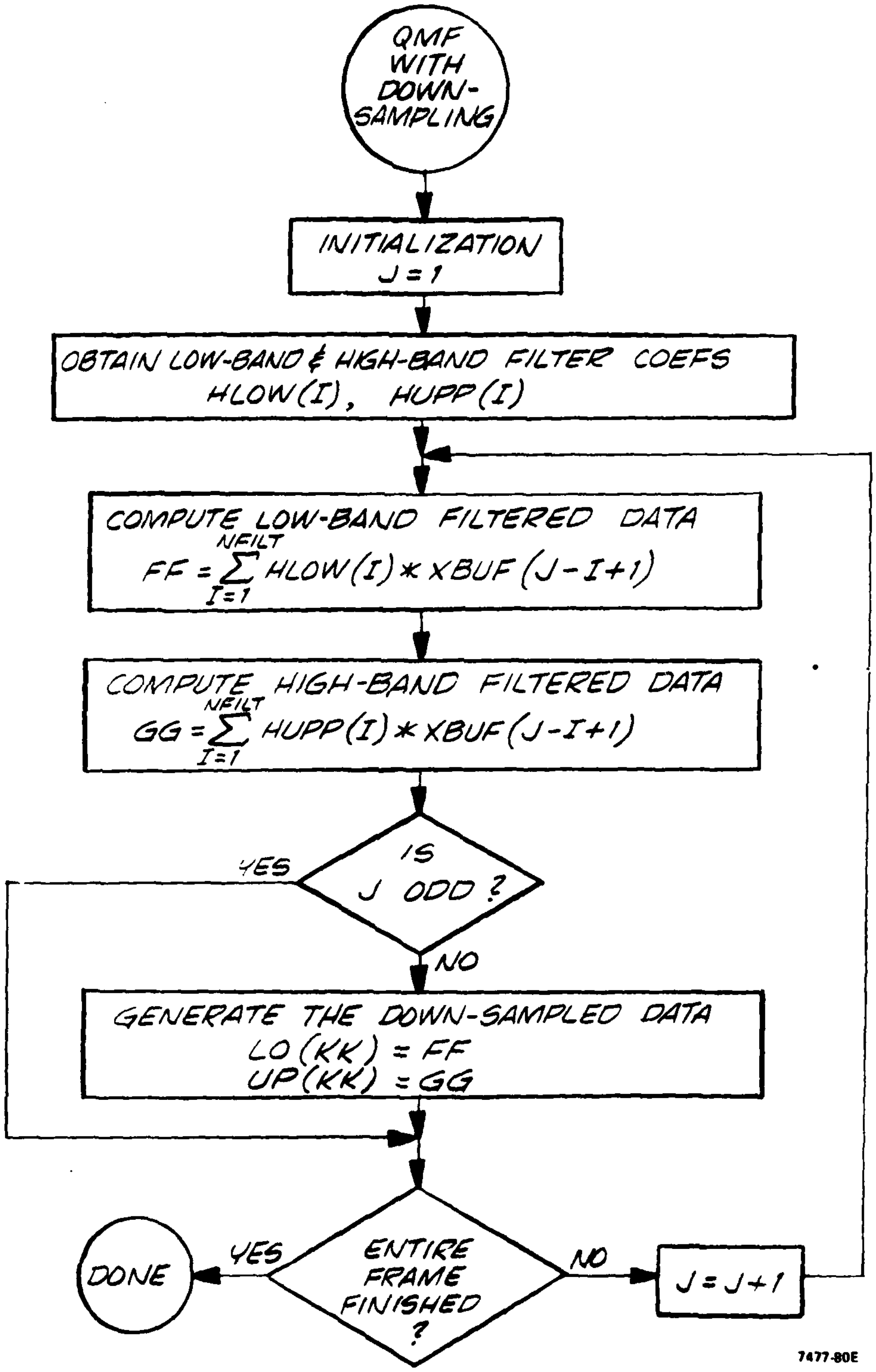


FIGURE 4.1.3 FLOW CHART OF QMF WITH DOWN-SAMPLING

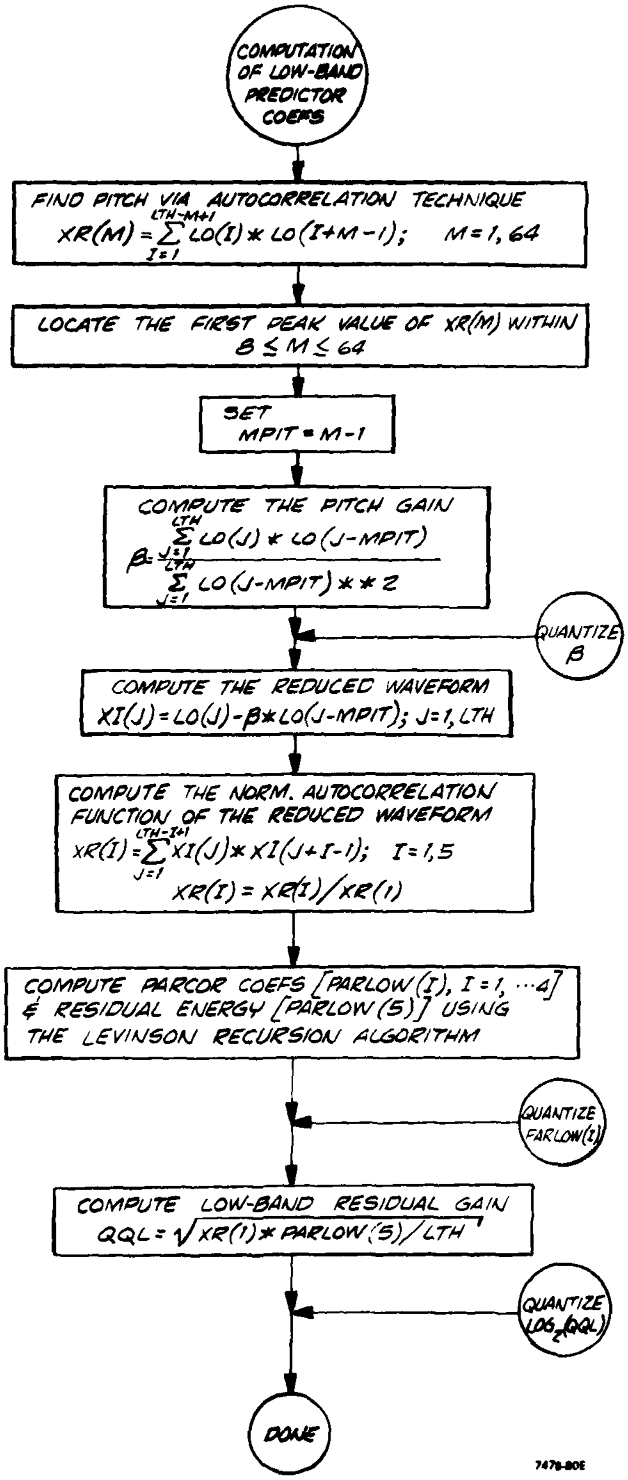


FIGURE 4.1.4 FLOW CHART OF LOW-BAND PREDICTOR COEFS COMPUTATION

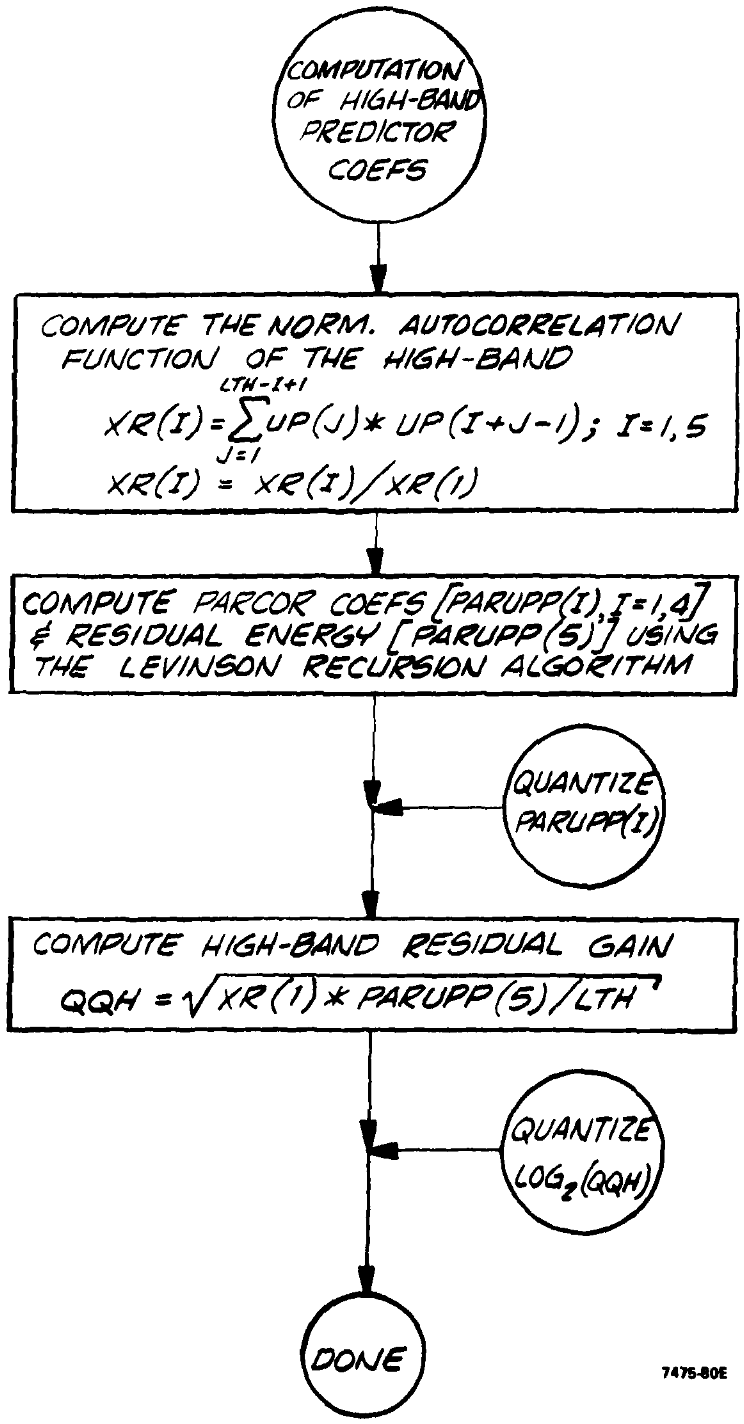


FIGURE 4.1.5 FLOW CHART OF HIGH-BAND PREDICTOR COEFS COMPUTATION

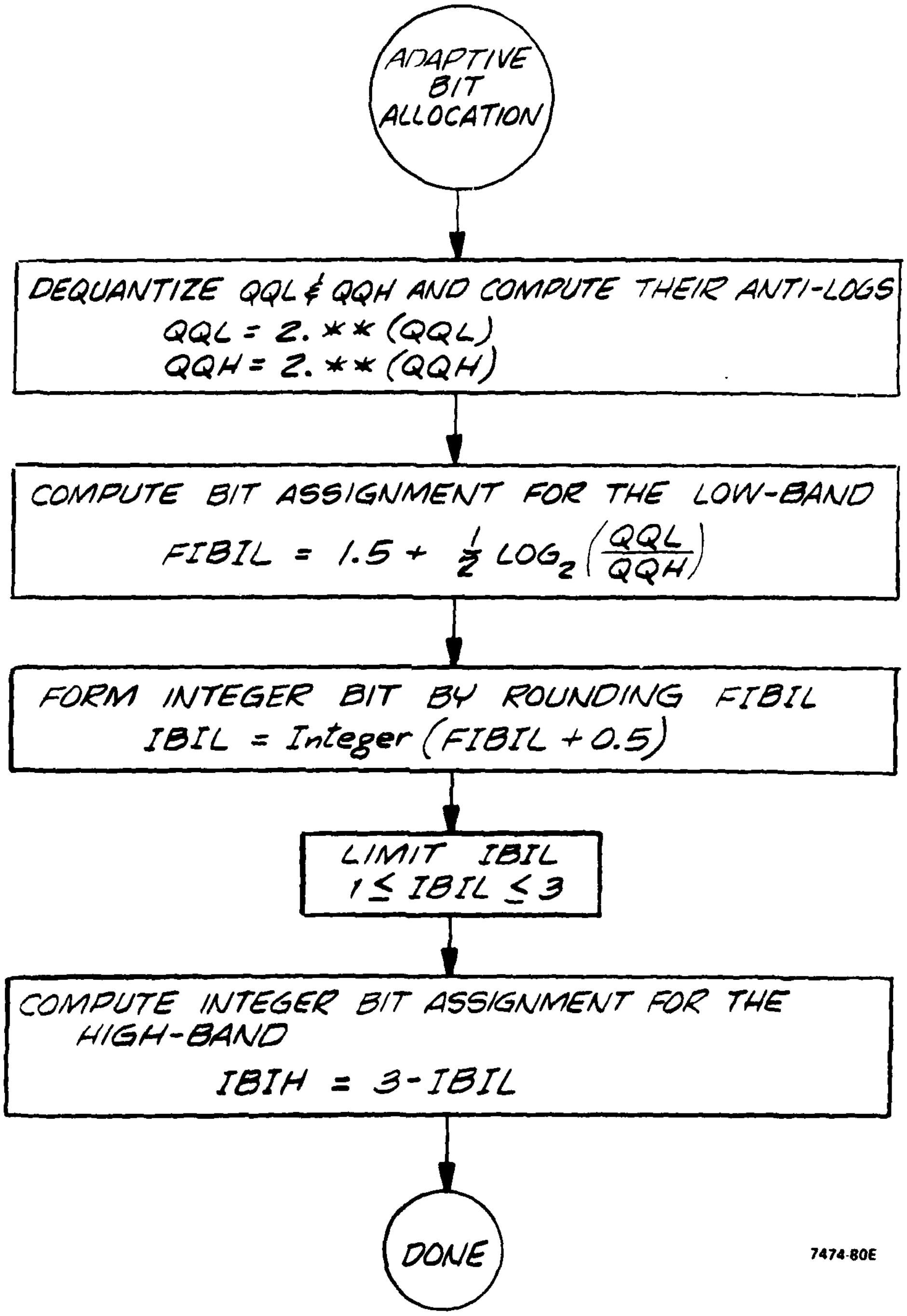


FIGURE 4.1.6 FLOW CHART OF ADAPTIVE BIT ALLOCATION

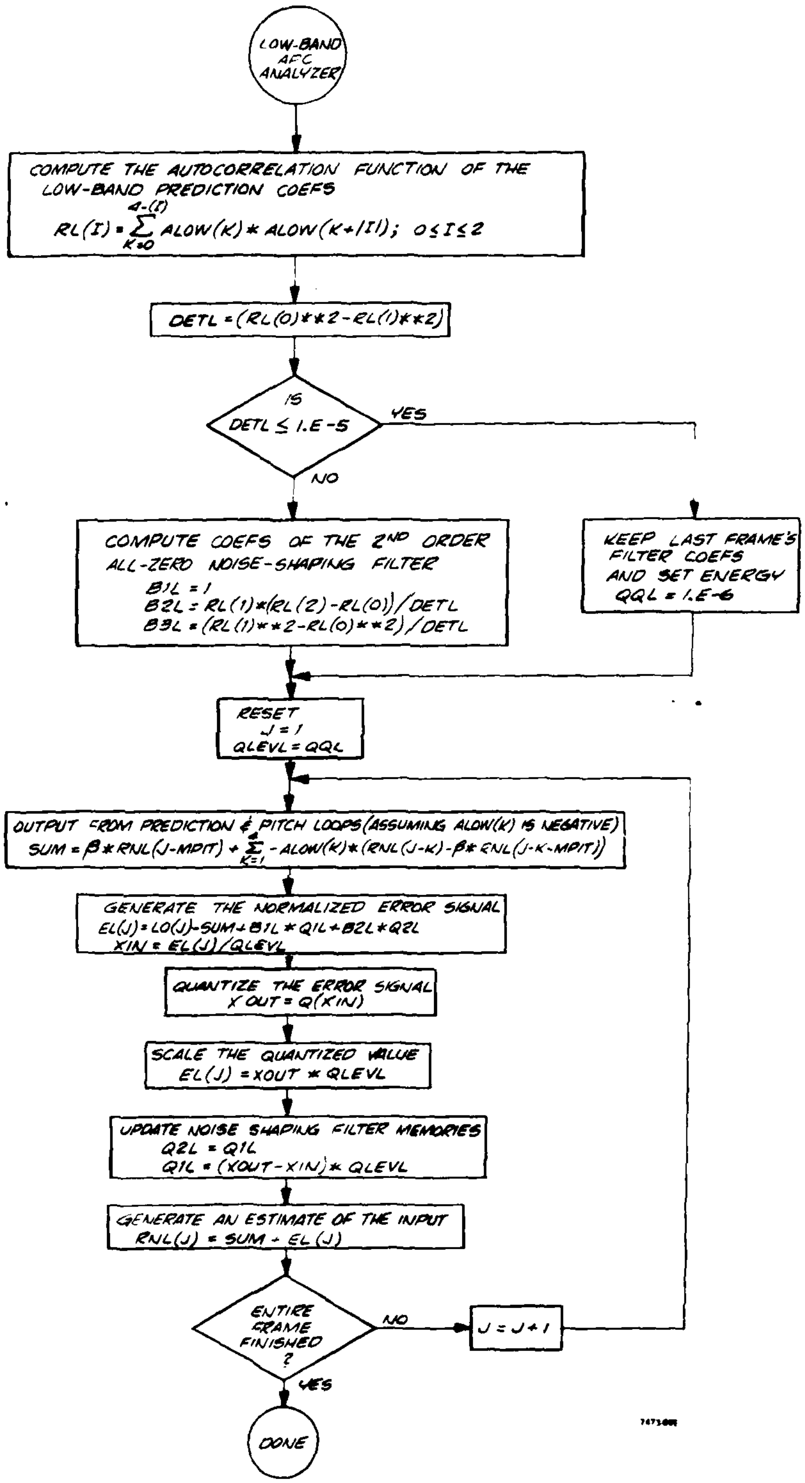


FIGURE 4.1.7 FLOW CHART OF LOW-BAND APC ANALYZER WITH NOISE SHARING

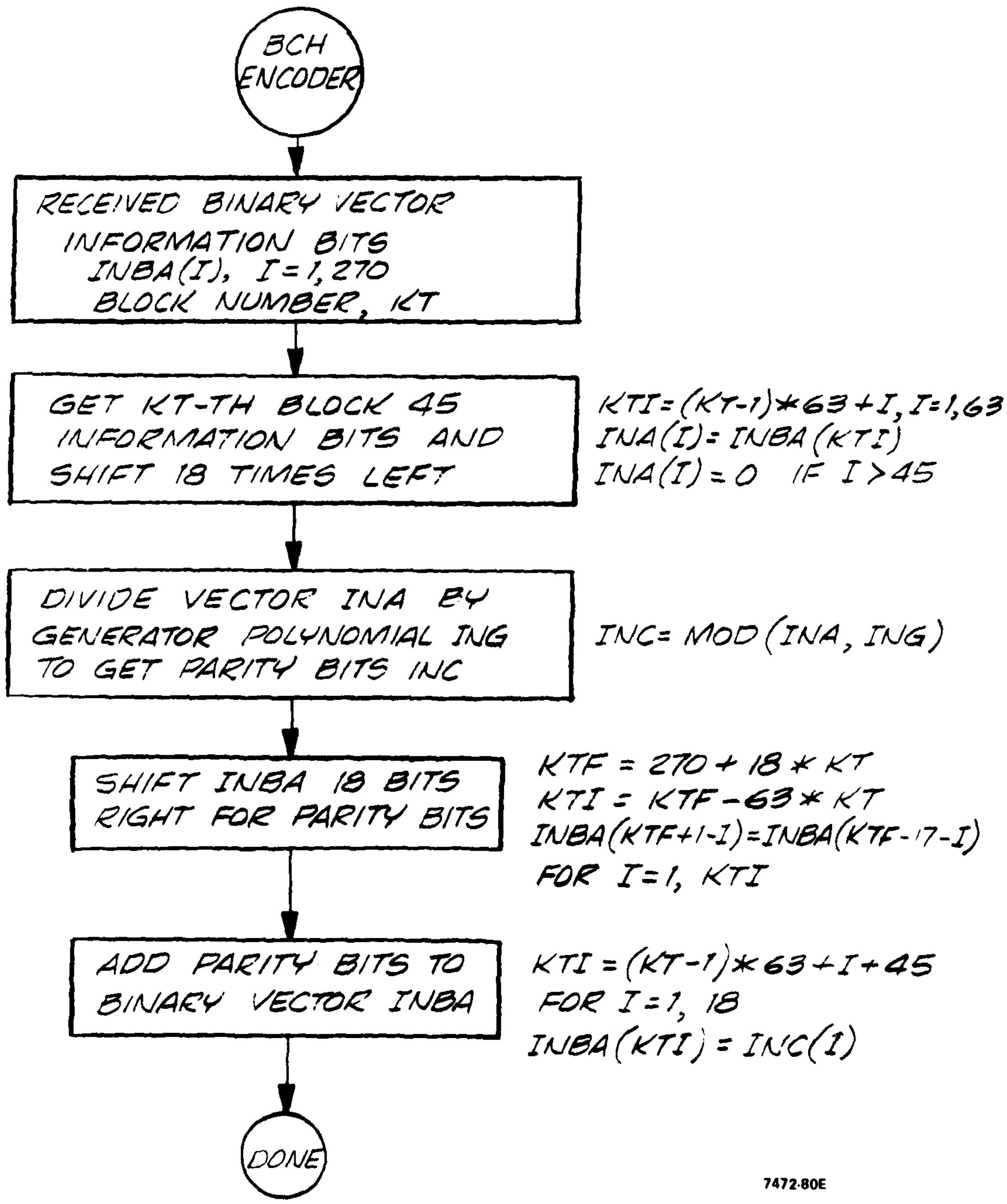


FIGURE 4.1.8 KT-TH BLOCK ENCODING ROUTINE FOR (63, 45) BCH CODE IN 16 KBPS SBAPC SYSTEM

deserialized back to APC parameters. Then the received residual signals are fed into the APC synthesizers as shown in Figure 4.1.9 , and the estimates of the low band and high band waveforms are generated. These sub-band signals are filtered using quadrature mirror filters as illustrated in Figure 4.1.10. The difference between the low and high band creates a replica of the input.

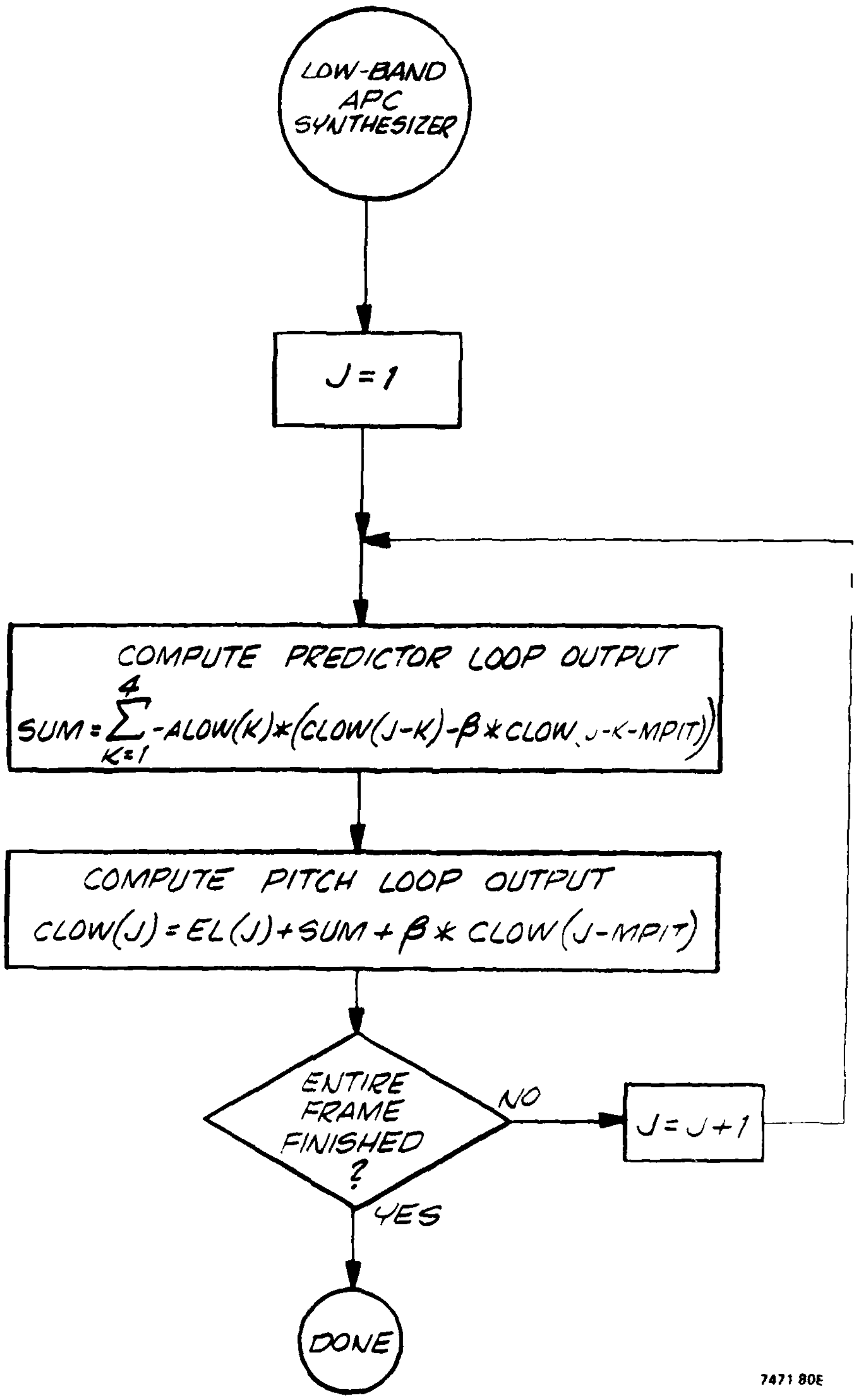


FIGURE 4.1.9 FLOW CHART OF LOW-BAND APC SYNTHESIZER

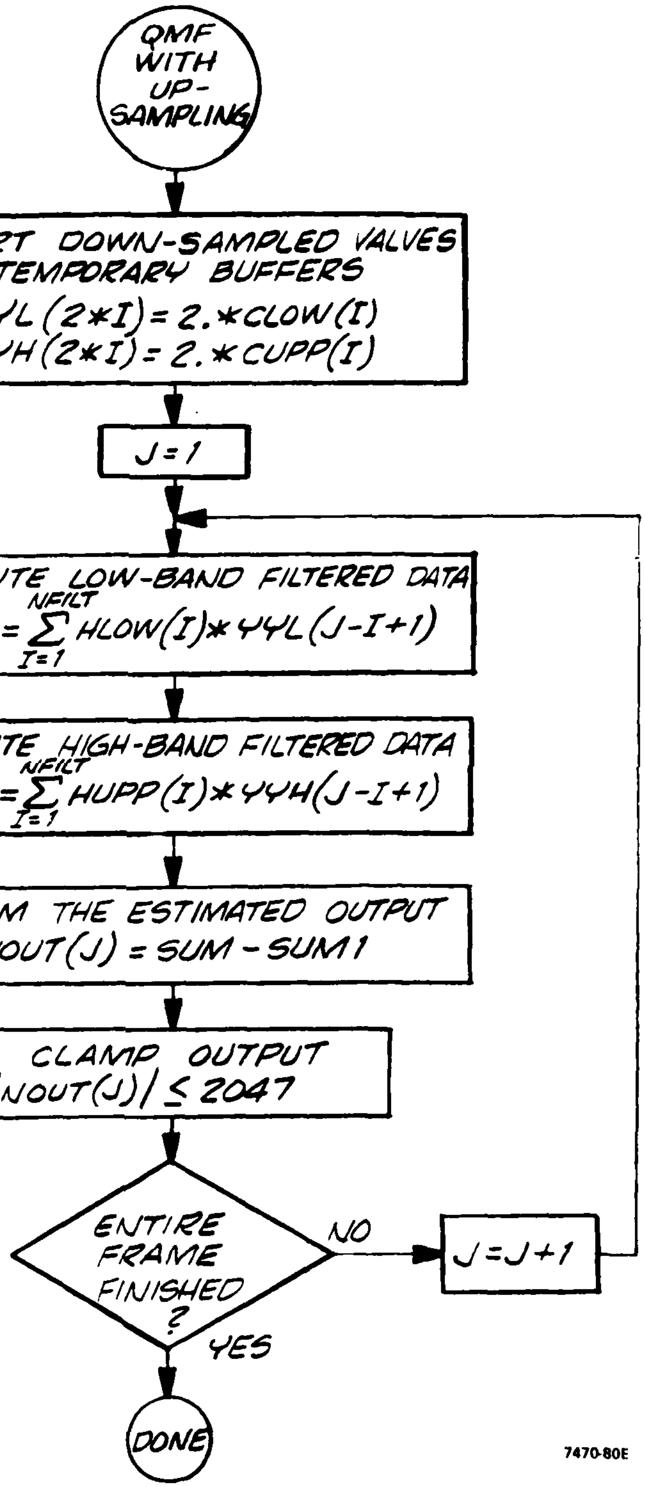


FIGURE 4.1.10 FLOW CHART OF QMF WITH UP-SAMPLING

4.2 The User's Guide

4.2.1 Task Building

To build the loadable module of the SBAPC program, issue the following indirect command:

@ SBAPC

Operations performed by this indirect command file includes the compilation of all FORTRAN routines, the purging of older FORTRAN and OBJECT modules, and the task building of TSK module. The print-outs involving the task building of the SBAPC program are shown as follows:

```
>@SBAPC
>!       SBAPC.CMD
>!       AUG. 11, 1980
>!       CMD FILE TO COMPILE AND BUILD SBAPC PROGRAM AT 16 KBPS
>PIP *.FTN/PU
>PIP *.OBJ;*/PU
>F4P SBAPC=SBAPC/NOTR
>F4P TAPE2=TAPE2/NOTR
>F4P FFTRR8=FFTRR8/NOTR
>F4P SER=SER/NOTR
>F4P CESR=CESR/NOTR
>F4P BNSR=BNSR/NOTR
>F4P DSER=DSER/NOTR
>F4P GF2AMD=GF2AMD/NOTR
>F4P CONV=CONV/NOTR
>PIP SBAPC.TSK;*/DE
>TKB SBAPC,LP=SBAPC,SER,CESR,BNSR,DSER,GF2AMD,CONV,TAPE2,FFTRR8

>@ <EOF>
>
```

4.2.2 Operating Procedures

After building the SBAPC.TSK module, the program can be started by issuing:

```
>RUN SBAPC
```

Print-outs of the actual running of the program for two situations (one with no noise suppression and one with noise suppression) are depicted in Figures 4.2.1 and 4.2.2.

```
RUN SBAPC

                    *** FORTRAN SIMULATION OF SBAPC ***

          ENTER PROGRAM PARAMETERS:
NOISE SUPPRESSION FACTOR[0:MIN,15:MAX]=8
CHANNEL ERROR RATE(E15.8)=1.E-2
BEGINNING FRAME NUMBER(I4)=    1
ENDING FRAME NUMBER(I4)=   10
SIGNAL-TO-NOISE COMPUTATION:   0=YES        1=NO   0
IS THE INPUT ON MAG. TAPE? N
IS THE OUTPUT GOING TO MAG TAPE? N
OUTPUT FILE NAME= OUT.DAT
INPUT FILE NAME= VOICE.3KC
FR #=   1    SNR= 0.2177E+02DB   CSNR= 0.2177E+02   CH. ERRS  0  1  0  0  0  0
FR #=   2    SNR= 0.1259E+02DB   CSNR= 0.1718E+02   CH. ERRS  1  0  0  1  2  0
FR #=   3    SNR= 0.8408E+01DB   CSNR= 0.1426E+02   CH. ERRS  0  1  0  1  1  0
FR #=   4    SNR= 0.8371E+01DB   CSNR= 0.1278E+02   CH. ERRS  2  0  2  2  0  0
FR #=   5    SNR= 0.1591E+02DB   CSNR= 0.1341E+02   CH. ERRS  2  3  1  0  0  1
FR #=   6    SNR= 0.1555E+02DB   CSNR= 0.1377E+02   CH. ERRS  1  0  0  1  0  0
FR #=   7    SNR= 0.1806E+02DB   CSNR= 0.1438E+02   CH. ERRS  0  1  0  0  0  0
FR #=   8    SNR= 0.1340E+02DB   CSNR= 0.1426E+02   CH. ERRS  1  0  0  0  0  0
FR #=   9    SNR= 0.1818E+02DB   CSNR= 0.1469E+02   CH. ERRS  0  0  0  1  0  0
FR #=  10    SNR= 0.1182E+02DB   CSNR= 0.1441E+02   CH. ERRS  0  0  0  0  0  0
MISSION ACCOMPLISHED
```

FIGURE 4.2.1: PRINTOUTS OF SBAPC PROGRAM (ξ=0, BER=0)

```
>RUN SBAPC

                    *** FORTRAN SIMULATION OF SBAPC ***

          ENTER PROGRAM PARAMETERS:
NOISE SUPPRESSION FACTOR[0:MIN,15:MAX]=0
CHANNEL ERROR RATE(E15.8)=0
BEGINNING FRAME NUMBER(I4)=     1
ENDING FRAME NUMBER(I4)=   10
SIGNAL-TO-NOISE COMPUTATION:   0=YES         1=NO    0
IS THE INPUT ON MAG. TAPE? N
IS THE OUTPUT GOING TO MAG TAPE? N
OUTPUT FILE NAME= OUT.DAT
INPUT FILE NAME= VOICE.3KC
FR #=   1    SNR= 0.1787E+02DB   CSNR= 0.1787E+02   CH. ERRS  0  0  0  0  0  0
FR #=   2    SNR= 0.1435E+02DB   CSNR= 0.1611E+02   CH. ERRS  0  0  0  0  0  0
FR #=   3    SNR= 0.1190E+02DB   CSNR= 0.1471E+02   CH. ERRS  0  0  0  0  0  0
FR #=   4    SNR= 0.9367E+01DB   CSNR= 0.1337E+02   CH. ERRS  0  0  0  0  0  0
FR #=   5    SNR= 0.1379E+02DB   CSNR= 0.1346E+02   CH. ERRS  0  0  0  0  0  0
FR #=   6    SNR= 0.1672E+02DB   CSNR= 0.1400E+02   CH. ERRS  0  0  0  0  0  0
FR #=   7    SNR= 0.1901E+02DB   CSNR= 0.1472E+02   CH. ERRS  0  0  0  0  0  0
FR #=   8    SNR= 0.1371E+02DB   CSNR= 0.1459E+02   CH. ERRS  0  0  0  0  0  0
FR #=   9    SNR= 0.1770E+02DB   CSNR= 0.1494E+02   CH. ERRS  0  0  0  0  0  0
FR #=  10    SNR= 0.1028E+02DB   CSNR= 0.1447E+02   CH. ERRS  0  0  0  0  0  0
MISSION ACCOMPLISHED
```

FIGURE 4.2.2: PRINTOUTS OF SBAPC PROGRAM (ξ=7, BER=10^{-2})

SECTION V

CONCLUSIONS AND RECOMMENDATIONS

5.1 Conclusions

This contract has resulted in the development of a high quality 16 Kb/s Split-Band Adaptive Predictive Coder (SBAPC) whose specifications are shown in Table 1-1. Based on our tradeoff analysis, the SBAPC system with separate pitch and short-term prediction loops performs better than the system with one combined loop. In the scheme with two loops, a 1st order pitch loop improves 2-3 dB signal-to-noise ratio as compared to the system without any pitch prediction. Though further improvement can be obtained with a 3rd order pitch predictor, the system is sometimes unstable at data rates below 12 KBPS. It is, therefore, recommended to use 1st order pitch prediction to always ensure the stability of the system. In contrast to the low band, the pitch information is not needed in the high band since this waveform contains little information about pitch. Our results also indicate that 4th order short-term predictors on both low and high bands represent a good compromise between the overhead bit rate and the quality of the processed speech. In addition, noise shaping algorithms have been found to be advantageous in SBAPC schemes. Particularly, Makhoul's second order all-zero noise shaping filter has resulted in slightly better quality speech as compared to that of the Atal's technique. As for the quantization of the residual signals, adaptive bit allocation of quantizer bits according to the energies of subbands yields further improvement in speech quality. Since these energies have to be sent to the receiver for quantization purposes, adaptive bit allocation does not require additional transmission of data.

Informal listening tests indicate that the SBAPC system yields much higher speech quality than that of CVSD in a back-to-back mode. Also, when compared with other high quality 16 Kb/s algorithms, such as, adaptive transfrom coding (ATC), the SBAPC processed speech is slightly low-passed, but its smooth quality is much preferred over that of ATC with the noticeable "dish-washing" background noise. Furthermore, the SBAPC system has also shown to perform well in simulated tactical situations. With the help of 5 blocks of (63,45) BCH codes, the algorithm yields high processed speech quality even in the presence of 10^{-2} channel error rate. In addition, the noise suppression routine in the SBAPC is capable of reducing background noise whose level is as high as -6 dB S/N which results in highly intelligible speech without the annoying noise components. Since the SBAPC is a modified version of adaptive predictive coding, it also tandems favorably with the 2.4 Kb/s linear predictive coder.

5.2 Recommendations

Based on our findings in this contract, the SBAPC algorithm can indeed replace the existing Continuously Variable Slope Deltamodulation scheme in future 16 Kb/s terminals. GTE strongly recommends that the SBAPC should be further studied and be implemented in real-time. In particular, the following areas should be pursued to enhance the performance and robustness of the algorithm:

1) segmental quantization of residual signals using pitch information
2) protection against channel errors with more efficient error-correcting codes
3) noise reduction with adaptive suppression factors

REFERENCES

1. D. Esteban and C. Galand, "Application of Quadrature Mirror Filters to Split-Band Voice Coding Schemes," Conference Record of IEEE International Conference on Acoustics, Speech and Signal Processing, pp. 191-195, Hartford, CT, May 1977.

2. D. Esteban and C. Galand, "32 KBPS CCITT Compatible Split-Band Coding Scheme," Conf. Record of IEEE International Conference Acoustics, Speech and Signal Processing, Tulsa, OK, April 1978.

3. A. Oppenheim and R. Schafer, Digital Signal Processing, Prentice Hall, 1975.

4. F. J. Harris, "On the Use of Windows for Harmonic Analysis with the Discrete Fourier Transform," Proc. IEEE, Vol. 66, No. 1, January 1978.

5. J. D. Johnston, "A Filter Family Designed for Use in Quadrature Mirror Filter Banks," Proc. IEEE Int. Conf. on Acoustics, Speech and Signal Processing, Denver, CO, April 1980.

6. B. Atal and M. Schroeder, "Adaptive Predictive Coding of Speech Signals," Bell System Tech. J., Vol. 48, October 1970, pp. 1973-1986.

7. A. J. Goldberg, R. L. Freudberg, and R. S. Cheung, "High Quality 16 Kb/s Voice Transmission," Conference Record of IEEE International Conference on Acoustics, Speech and Signal Processing, Philadelphia, PA, April 1976.

8. M. Berouti and J. Makhoul, "High Quality Adaptive Predictive Coding of Speech," Conference Record of IEEE International Conference on Acoustics, Speech and Signal Processing, Tulsa, OK, April 1978.

9. M. D. Paez and T. H. Glisson, "Minimum Mean Squared-Error Quantization in Speech PCM and DPCM Systems," IEEE Trans. Communications, Vol. COM-20, pp. 225-230, April 1972.

REFERENCES (Cont'd)

10. W. C. Adams, Jr. and C. E. Giesler, "Quantizing Characteristics for Signals Having Laplacian Amplitude Probability Density Function," IEEE Trans. Communications, Vol. COM-26, No. 8, August 1978.

11. J. Makhoul and M. Berouti, "Adaptive Noise Spectral Shaping and Entrophy Coding in Predicitve Coding of Speech," IEEE Trans. Acoustics, Speech and Signal Processing, Vol. ASSP-27, February 1979.

12. B. S. Atal and M. R. Schroeder, "Predictive Coding of Speech Signals and Subjective Error Criteria," IEEE Trans. Acoustic, Speech and Signal Processing, Vol. ASSP-27, June 1979.

13. N. S. Jayant, "Digital Coding of Speech Waveform: PCM, DPCM, and DM Quantizers," Proc. IEEE, Vol. 62, No. 5, May 1974, pp. 611-632.

14. J. L. Melsa and D. L. Cohn, Final Report: "Study of Sequential Estimation Methods for Speech Digitization," Contract No. DCA 100-74-C-0037, June 1975.

15. T. E. Tremain, et all, "Implementation of Two Real-Time Narrowband Speech Algorithms," Conference Record EASTCON 1978, pp. 698-708.

16. J. Max, "Quantizing for Minimum Distortion," IRE Trans. Inform. Theory, Vol. IT-6, pp. 7-12, 1960.

17. B. Widrow, et al, "Adaptive Noise Cancelling Principles and Applications," Proc. IEEE, Vol. 63, pp. 1692-1716, December 1975.

18. S. Boll, "Suppression of Acoustic Noise in Speech Using Spectral Subtraction," IEEE Trans. Acoustics, Speech and Signal Processing, Vol. ASSP-27, No. 2, April 1979.

REFERENCES (Cont'd)

19. R. J. McAuley and M. Malpass, "Speech Enhancement Using a Soft-decision Noisy Suppression Filter," IEEE Trans. Acoustics, Speech and Signal Processing, Vol. ASSP-28, No. 2, April 1980.

20. M. Sambur and N. Jayant, "LPC Analysis/Synthesis from Speech Inputs Containing Quantization Noise or Additive White Noise," IEEE Trans. Acoustics, Speech and Signal Processing, Vol. 24, No. 6, December 1976.

Appendix A

Theory of Quadrature Mirror Filters

One approach to band-split/reconstruct the input waveform is to make use of quadrature mirror filters (QMF) since the use of QMF design equations will achieve perfect splitting/reconstruction without large order filters. For explanatory purposes, consider the ideal splitting/reconstruction process described in Figure A.1. For this system, the following definitions apply:

a. $x(n)$ is a Nyquist band-limited signal with z-transform $X(z)$.

b. $h_1(n)$ is the impulse response of the low-pass filter the z-transform of which is $H_1(z)$.

c. $h_2(n)$ is the impulse response of the high-pass filter the z-transform of which is $H_2(z)$.

d. $y_1(n)$ is a baseband equivalent low-pass signal with z-transform $Y_1(z)$.

e. $y_2(n)$ is a baseband equivalent high-pass signal with z-transform $Y_2(z)$.

The signal, $x(n)$, is processed by filters $h_1(n)$ and $h_2(n)$ yielding the low-pass and high-pass equivalents, $x_1(n)$ and $x_2(n)$, of the input signal. As their spectra occupy half the Nyquist bandwidth of the original signal, the sampling rate in each band can be halved by decimating (ignoring) every second sample. For reconstruction, the signals $y_1(n)$ and $y_2(n)$ are interpolated, by inserting one zero-valued sample every other time, and then filtered respectively by $h_1(n)$ and $h_2(n)$ before being added, to give the signal $\hat{x}(n)$. The dashed lines, shown in Figure A.1 represent the data passed to the communication channel(s) by the speech processing system.

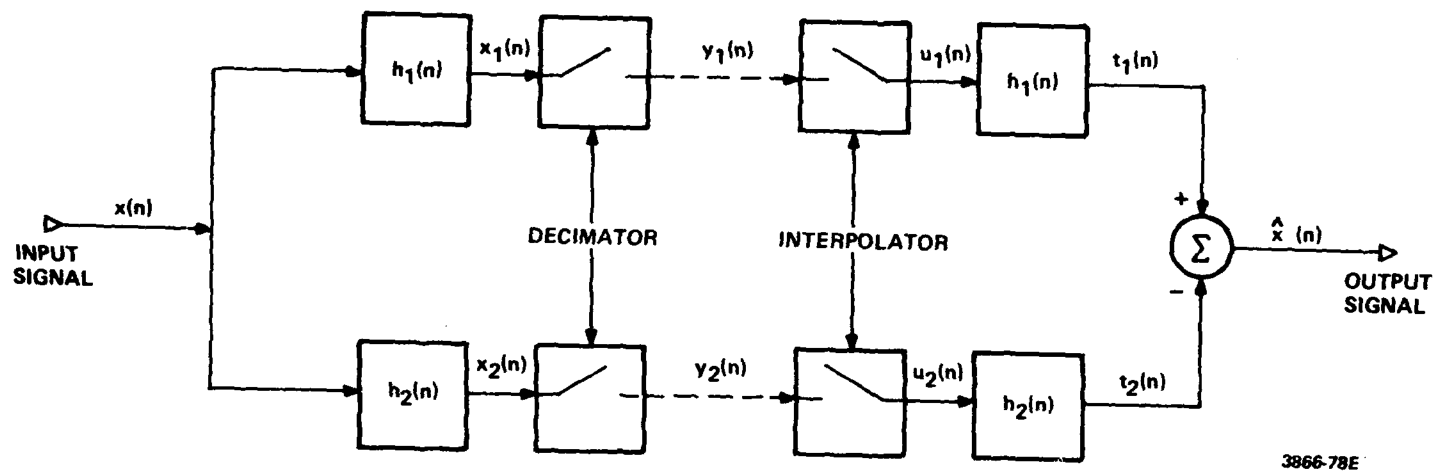


FIGURE A.1 BAND-SPLITTING AND RECONSTRUCTION USING QMF

In order to minimize $(\hat{x}(n) - x(n))$, certain restrictions on the filters, $h_1(n)$ and $h_2(n)$, must be met. We will derive these restrictions by constructing the transfer function of the QMF structure.

Using z-transform notation and referring to Figure A.1, we may write the intermediary filtered outputs as

$$X_1(z) = H_1(z) \quad X(z) \tag{A-1}$$

and

$$X_2(z) = H_2(z) \quad X(z) \tag{A-2}$$

The transforms of the decimated signals, $y_1(n)$ and $y_2(n)$, and of the interpolated signals, $u_1(n)$ and $u_2(n)$, are given by:

$$Y_1(z) = 1/2 \left[X_1(\tilde{z}) + X_1(-\tilde{z})\right] \qquad , \tilde{z} = z^{1/2} \tag{A-3}$$

$$Y_2(z) = 1/2 \left[X_2(\tilde{z}) + X_2(-\tilde{z})\right] \tag{A-4}$$

$$U_1(z) = Y_1(z^2) \tag{A-5}$$

$$U_2(z) = Y_2(z^2) \tag{A-6}$$

After the final filtering operation, the transforms of the reconstructed waveform components, $t_1(n)$ and $t_2(n)$, are given by

$$T_1(z) = H_1(z) \quad U_1(z) \tag{A-7}$$

$$T_2(z) = H_2(z) \quad U_2(z) \tag{A-8}$$

Using the relations expressed in (A-1) through (A-6), the z-transforms can be rewritten as

$$T_1(z) = 1/2 \left[H_1(z) \quad X(z) + H_1(-z) \quad X(-z)\right] \quad H_1(z) \tag{A-9}$$

$$T_2(z) = -1/2 \left[H_2(z) \quad X(z) + H_2(-z) \quad X(-z)\right] \quad H_2(z) \tag{A-10}$$

The z-transform of the reconstructed waveform, $\hat{x}(n)$, is obtained by adding (A-9) and (A-10)

$$\hat{X}(z) = 1/2 \left[H_1^2(z) - H_2^2(z)\right] X(z) + 1/2 \left[H_1(-z)\, H_1(z) - H_2(-z)\, H_2(z)\right] \quad X(-z) \tag{A-11}$$

If we assume that

$$H_2(z) = H_1(-z) \tag{A-12}$$

then the reconstructed waveform transform becomes

$$\hat{X}(z) = 1/2 \left[H_1^2(z) - H_1^2(-z)\right] X(z) \tag{A-13}$$

Evaluating z on the unit circle gives the Fourier transform of $\hat{X}(z)$

$$\hat{X}(e^{jwT}) = 1/2 \left[H_1^2(e^{jwT}) - H_1^2(e^{j(w + ws/2)\ T})\right] X(e^{jwT}) \tag{A-14}$$

For the case when $h_1(n)$ is an even, symmetrical FIR filter of order N, then it can be shown that (A-14) reduces to

$$\hat{X}(e^{jwT}) = 1/2\ e^{-j(N-1)wT}\ X(e^{jwT}) \tag{A-15}$$

where $H_1{}^2(e^{jwT})$ exhibits an odd symmetric property about $^{w}s/4$ and the half-power point $H_1{}^2(^{w}s/4) = 0.5$.

The inverse transform yields a perfectly reconstructed signal (no frequency distortion) with a gain factor of 1/2 and delay of N-1 samples as shown by

$$\hat{x}(n) = 1/2\ x(n - N + 1) \tag{A-16}$$

Therefore, we have shown that no guarantee perfect reconstruction of the original spectrum, the following filter constraints must be satisfied

$$h_1(n):\ \text{symmetrical, even order} \tag{A-17}$$

$$H_2(z) = H_1(-z) \tag{A-18}$$

$$H_1{}^2(e^{jwt}) + H_1{}^2(e^{j(w+\frac{ws}{2})T}) = 1 \tag{A-19}$$

APPENDIX B

Modified Robert's Noise Detection Algorithm

For all noise cancellation techniques, it is essential to obtain an accurate estimate of the statistics of the background noise. Hence, a noise detection algorithm is needed which considers only those frames of data which have a high probability of containing noise alone. In the modified Robert's algorithm, the boundary between noise and speech plus noise is established by monitoring the energy on a frame by frame basis and maintaining energy histograms which reflect the bimodal distribution (viz; one mode depicts the all noise state and one mode represents the speech plus noise state). The flow chart of the algorithm is shown in Figure B-1.

In this figure, the energy of the input speech is computed and normalized by a multiplication factor so that the maximum noise energy may vary around 32767. If the input energy does not exceed 16 bits (i.e., does not strongly imply the presence of speech), the algorithm updates the adaptive threshold. This routine first applies decay factor of 0.9944 to a 128-bin histogram of energy causing exponential decay of the histogram values with a time constant of 4 seconds. The value of the bin which encompasses the energy of the current frame is incremented by 144.

A second 128-point cumulative histogram is then formed to represent the area under the first histogram by computing the accumulated scores from the low energy bin to a high energy bin. If the 10th point of the second histogram exceeds 25% of the total area, it is assumed that there is no noise present (silence).

If noise is present, a search is made through the second histogram for the point which represents 80% of the total area. The quantum of energy

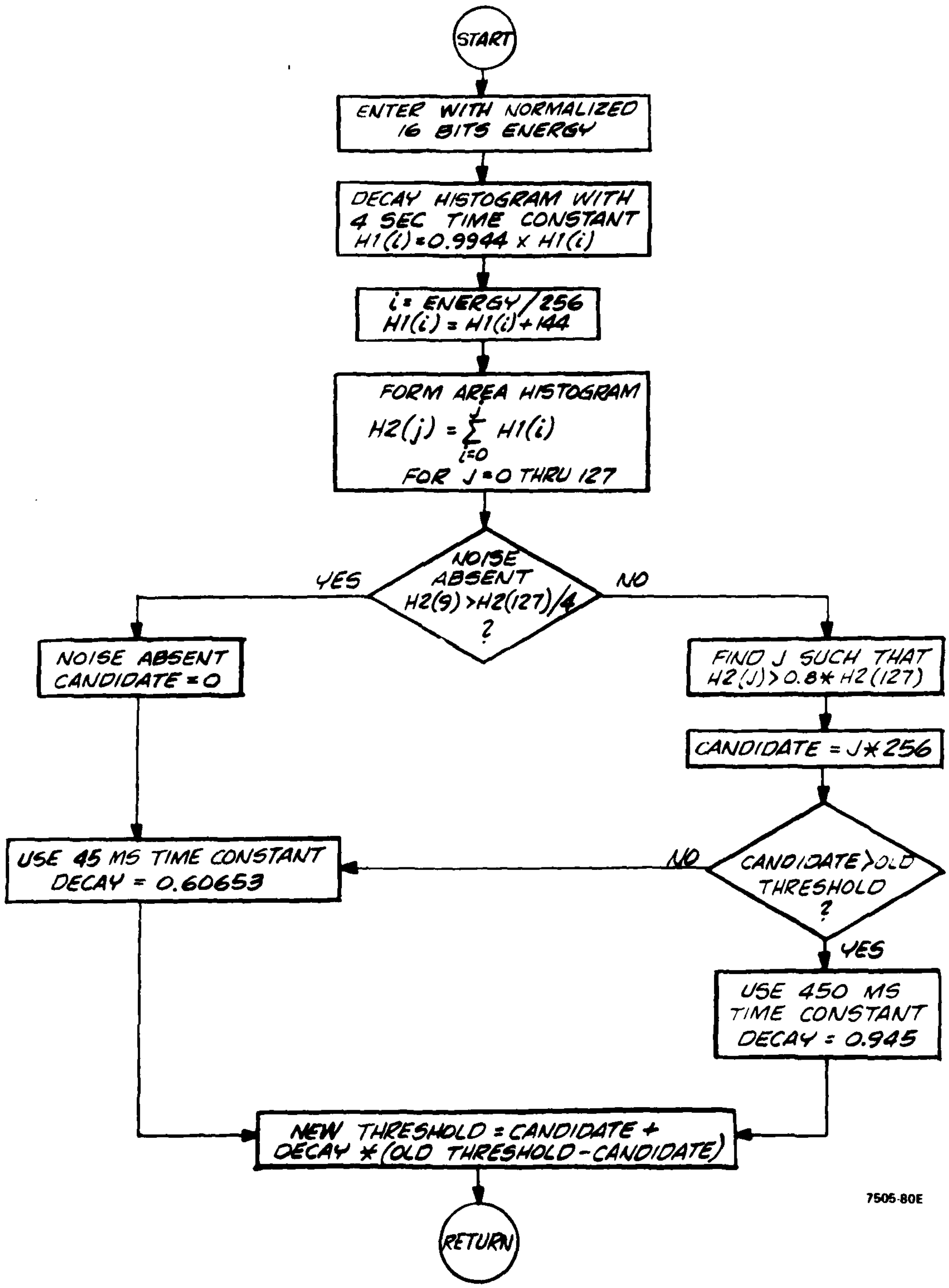


FIGURE B-1 MODIFIED ROBERT'S NOISE DETECTION ALGORITHM corresponding to this point becomes the new threshold candidate. If this candidate exceeds the current threshold, the threshold is updated using a decay factor of 0.945 (a slow time constant of 450 ms). If the candidate is below the current threshold, the threhsold is updated with a decay factor of .60653 (a fast time constant of 45 ms).

If noise is absent, the new threshold candidate is set to zero, and the threshold is updated using a decay factor of .60653 (a fast time constant of 45 ms). Finally, the threshold is held to a minimum of 1024 to guarantee updating of the estimated noise components when background noise suddenly disappears.

Appendix C Primitive BCH Codes

The BCH codes described in this appendix are cyclic codes that are well defined in terms of the roots of the generator polynomials [1]. These codes were discovered by Bose and Chaudhuri [2] - [3] and separately by Hocquenghem [4]. A binary (n,k) BCH code word consists of n symbols (bits in the binary case) where the first k bits are the information bits and the remaining r = n-k bits are redundant parity checks. It is convenient to represent code words with polynomials as

$$f(x) = f_0 + f_1 x + \dots + f_{n-1} x^{n-1}, \quad f_i \in GF(2) \tag{C-1}$$

where each bit position is associated with a locator. If f(x) is a code word, then

$$f_1(x) = f_1 + f_2 x + \dots + f_{n-1}{}^{n-2} + f_0 x^{n-1} \tag{C-2}$$

is also a codeword in a cyclic codes. In the primitive BCH code, which is the most convenient and powerful BCH code in theory and practice, the block length of the code may be defined as

$$n = 2^m - 1 \tag{C-3}$$

and with mt parity checks, it can correct any set of t independent errors within the block of n bits, where m and t are arbitrary positive integers [5]. This code may be described conveniently with the aid of finite Galois field theory introduced in Appendix D.

Let α be a primitive element of the finite field $GF(2^m)$, then the primitive BCH code may be described as the set of polynomials such that

$$f(\alpha^i) = 0, \; i = 1, 3, 5, \ldots, 2t - 1 \tag{C-4}$$

It is known in coding theory that these polynomials consist of all multiples of a single polynomial g(x), known as the generator polynomial. This polynomial also satisfies the equations as

$$g(\alpha^i) = 0, \; i = 1, 3, 5, \ldots, 2t - 1 \tag{C-5}$$

These generator polynomials are tabulated in Table C-1 for the selected primitive BCH codes.

Encoding Procedures

Let the k information bits be represented by the polynomial d(x) as

$$d(x) = \sum_{i=0}^{k-1} d_i \, x^i \tag{C-6}$$

then, the code word of n bits may be expressed as

$$f(x) = x^{n-k} \, d(x) + r(x) \tag{C-7}$$

where r(x) is the remainder (parity check) obtained according to the following equation:

$$\frac{x^{n-k} \, d(x)}{g(x)} = q(x) + \frac{r(x)}{g(x)} \tag{C-8}$$

Block length n	k	t	Generator Polynomial
63	57	1	$g_1(x) = (6, 1, 0) = x^6 + x + 1$
	51	2	$g_3(x) = g_1(x) \cdot (6, 4, 2, 1, 0)$
	45	3	$g_5(x) = g_3(x) \cdot (6, 4, 2, 1, 0)$
127	120	1	$g_1(x) = (7, 3, 0) = x^7 + x^3 + 1$
	113	2	$g_3(x) = g_1(x) \cdot (7, 3, 2, 1, 0)$
	106	3	$g_5(x) = g_3(x) \cdot (7, 4, 3, 2, 0)$
255	247	1	$g_1(x) = (8, 4, 3, 2, 0) = x^8 + x^4 + x^3 + x^2 + 1$
	239	2	$g_3(x) = g_1(x) \cdot (8, 6, 5, 4, 2, 1, 0)$
	231	3	$g_5(x) = g_3(x) \cdot (8, 7, 6, 5, 4, 2, 0)$

TABLE C-1 GENERATOR POLYNOMIALS FOR SELECTED PRIMITIVE BCH CODES where g(x) is the generator polynomial of the code. Therefore, encoding can be performed by the following procedures:

1). Calculate x^{n-k} d(x) by left shifting the information bits n-k times

2). Calculate the remainder (parity bits) r(x) from the division of x^{n-k} d(x) by g(x)

3). Add the polynomial x^{n-k} d(x) and r(x) to form the code word

The procedures of 1) and 3) can be done simply by shifting and addition. However, the procedure of 2) is rather involved in computation if the actual division is performed to get the remainder. If the BCH code is specified and it is desired to speed up the processing time of 2), it is recommended to use a look-up table procedure for the calculation of the remainder from 2). The code word is then transmitted through the noisy channel, where the received code word may be altered depending on the introduction of channel errors.

Decoding Procedures

There are several algorithms for a decoding of BCH codes. Efficient decoding algorithms have been discovered for BCH codes [1] - [7]. The Berlekamp decoder is particularly attractive for powerful codes that provide for a good deal of error corrections (e.g., 10 or more). The Peterson algorithm, however, is more efficient for less powerful codes (e.g., the codes used in generalized burst trapping). In this decoding procedure, the problem of finding efficient solutions to the key decoding equation will be addressed by using the Peterson technique.

When a BCH code word $\{f(x)\}$ is transmitted over a noisy channel, this code word may be corrupted by the channel, and what is received $\{\gamma(x)\}$ can be different from the intended code word. Thus, the received word may be expressed as

$$\gamma(x) = f(x) + e(x) \tag{C-9}$$

where $e(x)$ is the error polynomial which a decoder must compute to correct errors introduced by the channel. Let the received data be expressed in vector γ as

$$\gamma = [\gamma_0, \gamma_1, \ldots, \gamma_{n-1}] \tag{C-10}$$

or its associated polynomial $\gamma(x)$ by

$$\gamma(x) = \gamma_0 + \gamma_1 x + \ldots + \gamma_{n-1} x^{n-1} \tag{C-11}$$

Denote each of the error location numbers by β_j, $j = 1, 2, \ldots, t$, then it is shown [1] that the power sums S_i can be expressed as

$$S_i = \gamma(\alpha^i)$$

$$= \sum_{j=1}^{t} \beta_j^{\,i}, \qquad i = 1, 3, 5, \ldots, 2t-1 \tag{C-12}$$

In order to find the error locations, the Peterson procedures consist of three steps:

Step 1: Compute the power sums S_i from the received sequence through the relations

$$S_i = \gamma(\alpha^i),\ i = 1, 3, 5, \ldots, 2t-1$$

$$S_{2i} = S_i^2 \tag{C-13}$$

Step 2: Compute the symmetric functions σ_k, k = 1, 2, ..., t from the power sums S_i, i.e.,

$$\sigma(x) = x^t + \sigma_1 x^{t-1} + \ldots. + \sigma_{t-1}\, x + \sigma_t$$

$$= (x + \beta_1)(x + \beta_2) \ldots (x + \beta_t) \tag{C-14}$$

and the σ_k's may be obtained by the use of Newton's identities [1]

$$\underline{\sigma} = \begin{bmatrix} \sigma_1 \\ \sigma_2 \\ \cdot \\ \cdot \\ \cdot \\ \sigma_t \end{bmatrix}$$

$$= M_t^{-1}\, \underline{S} \tag{C-15}$$

$$= \begin{bmatrix} 1 & 0 & 0 & 0 & \ldots & 0 \\ S_2 & S_1 & 1 & 0 & \ldots & 0 \\ \cdot\cdot\cdot \\ \cdot\cdot\cdot \\ S_{2t-2} & S_{2t-3} & S_{2t-4} & & \ldots & S_{t-1} \end{bmatrix}^{-1} \begin{bmatrix} S_1 \\ S_3 \\ \cdot \\ \cdot \\ \cdot \\ S_{2t-1} \end{bmatrix}$$

If the determinant of M_t is singular, then reduce the error number t by 2 and proceed with it again.

Step 3: Find the error position locator β_j, $j = 1, 2, \ldots, t$, which is the roots of the polynomial $\sigma(x)$ in eq. (C-14).

An efficient algorithm for calculating the β_j's from eq. (C-14) has been developed by Chien [5], and all that remains to completely specify a binary BCH decoder is the computation of the coefficients of error locator polynomial, σ_j's. As it is noted from eq. (C-15), the calculation of the σ_j's involved matrix inversion which can be expressed analytically for the case $t \leq 3$. The results are:

For $t = 1$,

$$\sigma_1 = S_1$$

For $t = 2$,

$$\sigma_1 = S_1$$
$$\sigma_2 = (S_3 + S_1^3)/S_1$$

For $t = 3$,

$$\sigma_1 = S_1$$
$$\sigma_2 = (S_1^2 S_3 + S_5)/(S_1^3 + S_3)$$
$$\sigma_3 = (S_1^3 + S_3) + S_1\sigma_2$$

The calculation of the σ_i's and the estimation of the error number are shown in Figure A1 for t = 3. The flowchart of Chien's search decoding

$$m_1(\alpha) = \alpha^6 + \alpha + 1$$
$$= 0$$
$$= \alpha^{63} + 1$$

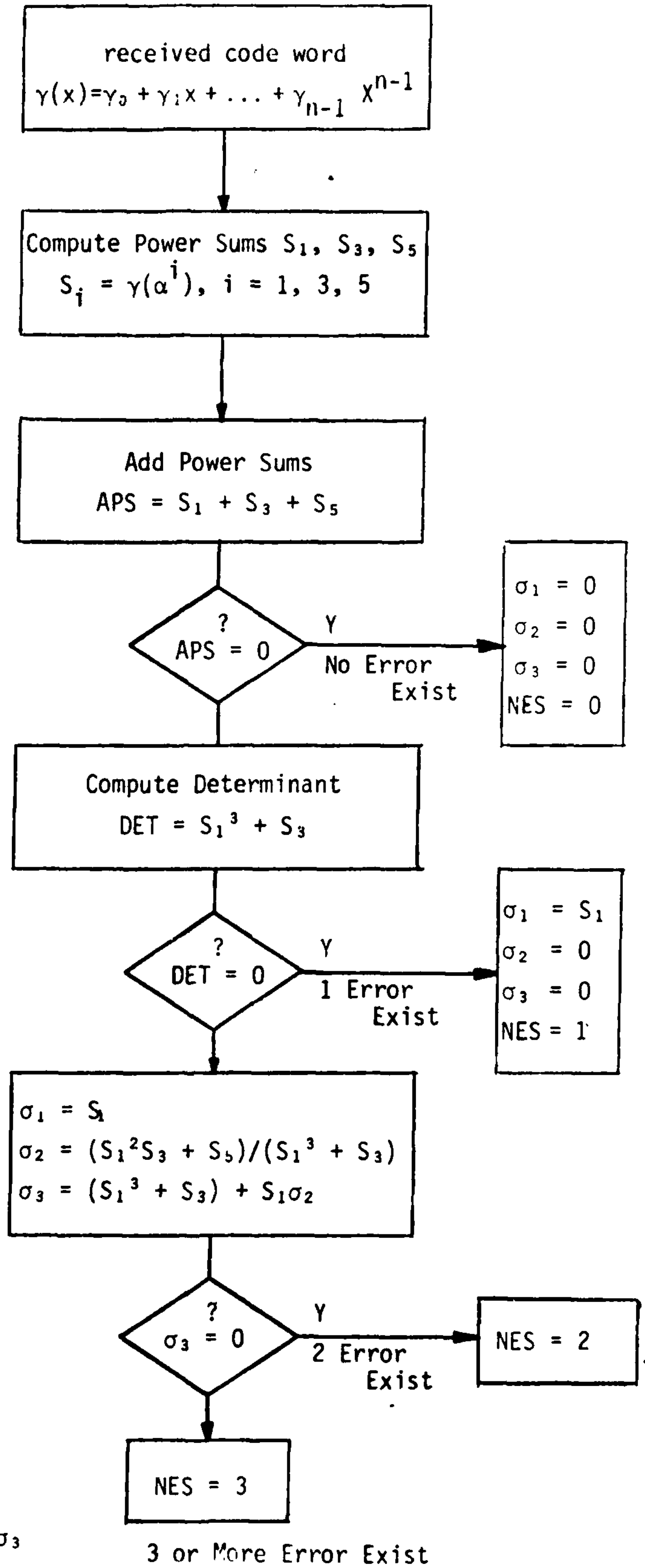


FIGURE C-1: COMPUTATION OF σ_1, σ_2, σ_3

procedure is shown in Figure C-2. This flowchart is for t = 3, i.e., the decoding algorithm can correct errors up to 3. One interesting observation in this decoding procedure is that the correction of errors may be performed erroneously if the number of errors in the block is greater than 3. Hence, the corrections may introduce additional channel errors. In order to avoid these additional errors, error corrections are made only when the estimated error number (NES in FigureC-1) equals to the measured error number (K in Figure C-2). This procedure eliminates most of the additional errors when more than 3 errors exist in the received word. In other words, the detection of errors more than 3 (i.e., 4, 5, 6, ..., etc.) is feasible in most of the cases. This fact contributes some improvements of the coder performance when the channel is very noisy (bit error rate $\approx 10^{-2}$).

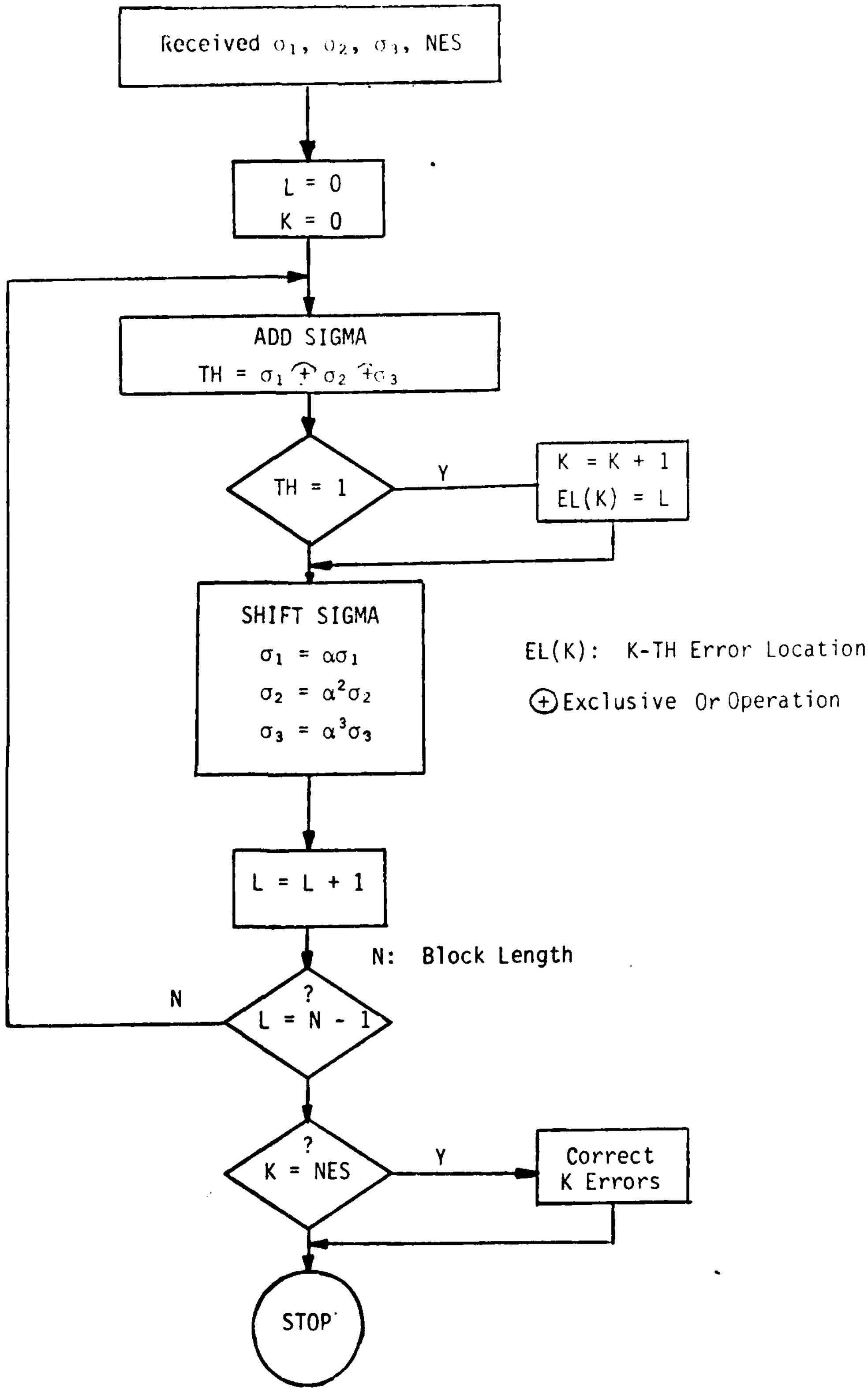


FIGURE C-2: CHIEN'S SEARCH DECODING PROCEDURE

References for Appendix C

[1] Peterson, W. W., Error-Correcting Codes, The MIT Press, Cambridge, MA, 1961.

[2] Bose, R. C. and D. K. Ray-Chandhuri, "On a Class of Error-Correcting Binary Group Codes," IEEE Trans. on Information and Control, Vol. IC-3, pp. 68-79, 1960.

[3] Bose, R. C. and D. K. Ray-Chandhuri, "Further Results on Error-Correcting Binary Group Codes," IEEE Trans. on Information and Control, Vol. IC-3, pp. 279-290, 1960.

[4] Hocquenghem, A., "Codes Correcteurs d'erreurs," Chiffres, Vol. 2, pp. 147-156, 1959.

[5] R. T. Chien, "Cyclis Decoding Procedures for Bose-Chandhuri-Hocquenghem Codes," IEEE Trans. on Information Theory, Vol. IT-10, pp. 357-363, 1964.

[6] E. R. Berlekamp, Algebraic Coding Theory, New York, McGraw-Hill, 1968.

[7] J. L. Massey, "Shift-Register Synthesis and BCH Decoding," IEEE Trans. on Information Theory, Vol. IT-15, No. 1, pp. 122-127, 1969.

Appendix D Operations in Galois Field

A Galois field is a finite set of elements that satisfy the axioms of a general field. Two operations (addition and multiplication) and their inverses are defined on the field elements. There is an identity element for each field element for both of the operations (0, 1) that is itself in the field. Also, both addition inverses and multiplication inverses are in the field. Finally, the rules of commutation and associativity are obeyed by the elements of the field.

Consider the following sixteen polynomials and their vector binary representations.

0	0000
1	0001
$1 + X$	0011
$1 + X + X^2$	0111
$1 + X + X^2 + X^3$	1111
X	0010
$X + X^2$	0110
$X + X^2 + X^3$	1110
X^2	0100
$X^2 + X^3$	1100
X^3	1000
$1 + X^3$	1001
$1 + X^2$	0101
$1 + X^2 + X^3$	1101
$X + X^3$	1010
$1 + X + X^3$	1011

As long as addition and multiplication of these polynomials is defined so that the axioms for the field are obeyed, then this will, in fact, be a Galois field of 2^4 elements ($GF(2^4)$).

Addition is defined to be modulo 2. Each element is its own additive inverse and addition and subtraction of elements are the same.

Multiplication must be defined so that the product of two elements does not take us out of the field. For this reason, multiplication in a Galois field is not ordinary multiplication of polynomials. Rather, multiplication is defined modulo an irreducible polynomial, the primitive polynomial of the Galois field. For our field $GF(2^m)$, the primitive polynomial is $1 + X + X^4$. To generate the 16 vectors in the field, all one needs to do is to divide X^m where m = 0, 1, ... 14 by the primitive polynomial.

0	-1	
1	0	
X	X	
X^2	X^2	
X^3	X^3	
X^4	$1 + X$	$1 + X + X^4 \overline{)\, X^4}$ quotient 1, $R\left[1 + X\right]$
X^5	$X + X^2$	$1 + X + X^4 \overline{)\, X + X^5}$ quotient X, $R\left[X + X^2\right]$
X^6	$X^2 + X^3$	
X^7	$X^3 + X + 1$	
X^8	$X^2 + 1$	
X^9	$X^3 + X$	
X^{10}	$X^2 + x + 1$	
X^{111}	$X^3 + X^2 + 1$	
X^{12}	$X^3 + X^2 + X + 1$	
X^{13}	$X^3 + X^2 + 1$	
X^{14}	$X^3 + 1$	
X^{15}	$X^0 + 1$	

It is now seen that the product of two binary vectors in the field is just the sum of their powers. The table repeats every fifteen powers so it is all done modulo 15.

$$X^i + X^j = X^{i+j} \pmod{15}$$

$$\frac{X^i}{X^j} = X^{i-j} \pmod{15}$$

APPENDIX E

LISTINGS OF FORTRAN PROGRAMS

```
!       SBAPC.CMD
!       AUG. 11, 1980
!       CMD FILE TO COMPILE AND BUILD SBAPC PROGRAM AT 16 KBPS
PIP *.FTN/PU
PIP *.OBJ;*/PU
F4P SBAPC,LP:=SBAPC/NOTR
F4P TAPE2,LP:=TAPE2/NOTR
F4P FFTRR8,LP:=FFTRR8/NOTR
F4P SER,LP:=SER/NOTR
F4P CESR,LP:=CESR/NOTR
F4P BNSR,LP:=BNSR/NOTR
F4P DSER,LP:=DSER/NOTR
F4P GF2AMD,LP:=GF2AMD/NOTR
F4P CONV,LP:=CONV/NOTR
PIP SBAPC.TSK;*/DE
TKB SBAPC,LP=SBAPC,SER,CESR,BNSR,DSER,GF2AMD,CONV,TAPE2,FFTRR8
```

```
      C       @@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
      C       @                                  @
      C       @            SBAPC.FTN             @
      C       @                                  @
      C       @@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
      C
      C
      C       FORTRAN PROGRAM TO SIMULATE THE SPLIT-BAND ADAPTIVE
      C       PREDICTIVE CODING(SBAPC) AT 16 KBPS
      C       STUDIED FOR DCA UNDER CONTRACT 100-79-C-0038
      C       BY GTE PRODUCTS INC., NEEDHAM HEIGHTS, MASS.02194
      C       DATE :OCT-6-80
      C
      C
      C
      C
      C
      C       TO BUILD THE TSK FILE, TYPE
      C
      C              @SBAPC
      C
      C       TO RUN THE PROGRAM, TYPE
      C
      C              RUN SBAPC
      C
      C       DEVICE ALLOCATIONS:      1=SBAPC PARAMETER FILE
      C                                2=INPUT FOR TAPE2
      C                                3=OUTPUT FOR TAPE2
      C                                4=CHANNEL ERROR FILE
      C                                5=KEY BOARD
      C                                6=LINE PRINTER
      C
      C
              COMMON/MTAPE0/NIN(170),NOUT(170)
              COMMON/MTAPE1/NSKIP,IST,NTOTI,NTUPS,NTOTO
              COMMON/MTAPE2/NEND,NERR,NFILE,NINS,NOUTS
              COMMON/TBL/QTBL(457)
              COMMON/NSTBL/FNSTBL(50)
              COMMON/SBTA/IBPT,IBBT,IBQL,IBQH,IBPL(4),IBPH(4)
              COMMON/SDTA/MPIT,IBETA,IQQL,IQQH,IDPL(4),IDPH(4)
              COMMON/SW/ICOUNT,IPRSW
              DIMENSION XR(256),XI(256)
              DIMENSION HLOW(32),HUPP(32)
              DIMENSION YYH(288),YYL(288),CLOW(144),CUPP(144)
              DIMENSION PARLOW(5),PARUPP(5),ALOW(5),AUPP(5)
              DIMENSION RNL(144),RNH(144)
              DIMENSION XBUF(288),EL(72),EH(72)
              DIMENSION INBA(360),INB(63),NERB(6),INC(18)
              REAL OLP(32),LO(144),UP(144)
      C       BITS ALLOCATIONS TO QUANTIZE SIDE INFORMATION
              DATA IBPT,IBBT,IBQL,IBQH,IBPL,IBPH/6,4,4*5,3,3,4,4,3,3/
      C*****
      C       I-O INTERACTION
      C*****
      C
      70      IKB=5
```

```
0019                IWR=6
          C         PRINT INTERMEDIATE RESULTS:     0=NO     1=PRINT
0020                ISW0=0
0021                ISW1=0
0022                IPRSW=0
          C
          C
0023                WRITE(IKB,200)
0024      200       FORMAT(///,20X,'*** FORTRAN SIMULATION OF SBAPC ***',
                    1//,10X,'ENTER PROGRAM PARAMETERS:')
          C
          C
          C         NOISE SUPRRESION FACTOR
          C
0025                WRITE(IKB,217)
0026      217       FORMAT(1H$,'NOISE SUPPRESSION FACTOR[0:MIN,15:MAX]=')
0027                READ(IKB,216)NSF
          C
          C         CHANNEL ERROR RATE
          C
0028                WRITE(IKB,213)
0029      213       FORMAT(1H$,'CHANNEL ERROR RATE(E15.8)=')
0030                READ(IKB,490)CERT
          C
          C
0031                WRITE(IKB,214)
0032      214       FORMAT(1H$,'BEGINNING FRAME NUMBER(I4)= ',3X)
0033                READ(IKB,216)LCOUNT
0034                WRITE(IKB,215)
0035      215       FORMAT(1H$,'ENDING FRAME NUMBER(I4)=',3X)
0036                READ(IKB,216)MCOUNT
0037      216       FORMAT(I6)
          C
0038                WRITE(IKB,242)
0039      242       FORMAT(1H$,'SIGNAL-TO-NOISE COMPUTATION:
                    1 0=YES          1=NO',3X)
0040                READ(IKB,246)ISNR
0041      246       FORMAT(I1)
          C
          C
          C
          C
          C
          C*****
          C         INITIALIZATION
          C*****
0042                LX=144
0043                NOL=18
0044                XP=FLOAT(NOL+1)
0045                LOCNT=1
0046                NFILT=32
0047                NFILT2=NFILT/2
0048                ILMX=3
0049                ILMN=1
0050                IQIL=406
0051                IQIH=432
```

```
              IRN=0
              JRN=0
              CSNR=0.
              KOUNT=0
              XSNR=0.
              Q1L=0.0
              Q2L=0.0
              Q1H=0.0
              Q2H=0.0
              DO 302 I=1,LTH
              RNL(I)=0.
              RNH(I)=0.
              LO(I)=0.
      302     UP(I)=0.
              DO 303 I=1,LX
              XBUF(I)=0.
              YYL(I)=0.
              YYH(I)=0.
              CUPP(I)=0.
      303     CLOW(I)=0.
              ICOUNT=0
      C
      C
      C*****
      C       MAG TAPE OR DISK I-O
      C*****
              NEND=0
              LTH=LX/2
              FLTH=LTH
              NTOTI=LX+NOL
              NTOTO=LX
              NTUPS=LX
              IST=1
              NSKIP=1
              NSKIPS=NSKIP
              NFILE=1
              CALL TAPE2(8)
      C
      C
      C*****
      C       READ IN PARAMETERS FOR FILTERS & QUANTIIZERS
      C*****
              CALL ASSIGN(1,'PARAM.DAT')
              DO 495 I=1,NFILT
      495     READ(1,490)HLOW(I)
      490     FORMAT(E15.8)
      C
      C
      C       INITIALIZE RANDOM NUMBER GENERATORS
              DO 491 I=1,457
              CALL RANDU(IRN,JRN,YQ)
              READ(1,493)QTBL(I)
      493     FORMAT(1X,E12.5)
      491     CONTINUE
      C
      C       READ NOISE SUPPRESSION TABLE
```

```
        C
0093              DO 497 I=1,NSF
0094              READ(1,493)(FNSTBL(J),J=1,50)
0095    497       CONTINUE
0096              CALL CLOSE(1)
        C
        C
        C
        C
        C*****
        C         DEFINITION OF SBAPC SYSTEM PARAMETERS
        C*****
0097              NLOW=4
0098              NUPP=4
0099              NNN=NLOW+1
0100              NNU=NUPP+1
        C
        C
        C***
        C         COMPUTE THE HIGH BAND IMPULSE RESPONSE
        C***
0101              DO 144 I=1,NFILT
0102              HUPP(I)=HLOW(I)*(-1)**(I-1)
        D         WRITE(IKB,143)HLOW(I),HUPP(I)
        D143      FORMAT(1X,'HLOW-HUPP',2(E15.8))
0103    144       CONTINUE
        C
        C
        C++++++++++++++++++++++++++++++++++++++++++++++
        C
        C         TRANSMITTER STARTS HERE
        C
        C++++++++++++++++++++++++++++++++++++++++++++++
        C
        C
        C*****
        C         DATA INPUT
        C*****
0104    1000      ICOUNT=ICOUNT+1
0105              CALL TAPE2(1)
0106              IF (NERR .NE. 0)GOTO 4900
        C         IF THE END OF TAPE IS REACHED,PROCESS 1 MORE FRAME
0107              IF(NEND .EQ. 0)GOTO 1014
0108              DO 1005 I=1,LX
0109    1005      NIN(I)=0
0110    1014      IF(ICOUNT .LT. LCOUNT)GOTO 1000
0111              IF(ICOUNT .GT. MCOUNT)GOTO 5000
        C
        C
0112              DO 1015 J=1,LX
0113              XBUF(LX+J)=NIN(J)
0114    1015      CONTINUE
        C
        C*****
        C         SUPPRESS BACKGROUND NOISE IF DESIRED
        C*****
```

```
              C
0115                    IF(NSF.EQ.0)GO TO 8100
              C
              C         PASS THROUGH THE NOISE SUPPRESSION FILTER
              C
0116                    CALL MRNSA(XR,XI,NTOTI,NSF)
              C
0117                    DO 8040 I=1,NTOTI
0118                    XBUF(LX+I)=XR(I)
0119          8040      CONTINUE
              C
              C         INTERPOLATE FRAME  BOUNDARY
              C
0120                    DO 8020 I=1,NOL
0121                    F1=FLOAT(I)/XP
0122                    F2=1.-F1
0123                    XBUF(LX+I)=XBUF(LX+I)*F1+F2*OLP(I)
0124          8020      OLP(I)=XR(LX+I)
              CCC
              C
0125          8100      CONTINUE
              C***
              C         BANDPASS FILTERING WITH QUADRATURE MIRROR FILTERS
              C***
0126                    DO 1101 J=1,LX
0127                    FF=0.
0128                    GG=0.
0129                    DO 1100 I=1,NFILT
0130                    FF=FF+HLOW(I)*XBUF(LX+J-I+1)
0131                    GG=GG+HUPP(I)*XBUF(LX+J-I+1)
0132          1100      CONTINUE
0133                    KK=J/2
0134                    JJ=KK*2
0135                    IF(JJ .NE. J)GOTO 1101
0136                    LO(LTH+KK)=FF
0137                    UP(LTH+KK)=GG
0138          1101      CONTINUE
              C
              C
              C         @@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
              C         @                                      @
              C         @         LOW BAND ENCODING            @
              C         @                                      @
              C         @@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
              C
              C
              C         COMPUTE THE ACF OF INPUT
              C
0139                    DO 1105  M=1,64
0140                    XR(M)=0.
0141                    LTR=LTH-M+1
0142                    DO 1105 I=1,LTR
0143          1105      XR(M)=XR(M)+LO(LTH+I)*LO(LTH+I+M-1)
              C
              C
              C         FIND PITCH NUMBER SKIPPING UNVOICED FRAMES
```

```
      C
                TERM=0.0
                TERM1=0.0
                XLG=0.0
                MPIT=0
                ISLOPE=1
                IF(IPITL.EQ.1.OR.XR(1).LE.1)GO TO 2047
                DO 2040 I=8,64
                IF(XR(I) .LT. 0)ISLOPE=-1
                IF(ISLOPE.EQ. 1.OR.XR(I).LE.XLG)GOTO 2040
                MPIT=I-1
                XLG=XR(I)
      2040      CONTINUE
      C
      C
      C*****
      C         CALCULATE FEEDBACK GAIN IN PITCH LOOP
      C*****
                DO 2045 J=1,LTH
                TERM1=TERM1+LO(LTH+J)*LO(LTH+J-MPIT)
      2045      TERM=TERM+LO(LTH+J-MPIT)**2
      2047      BETA=0.
                IF(TERM .NE.0)BETA=TERM1/TERM
      C
      CDDDDDDDDDDDDDDDDDDDDDD
      C
      D         WRITE(IKB,2048)IBBT,BETA,IBETA
      C
      CDDDDDDDDDDDDDDDDDDDDDD
      C
      C
      C         QUANTIZE BETA
                IQI=1
                CALL QTZ(IQI,IBBT,BETA,IBETA)
      C
      CDDDDDDDDDDDDDDDDDDD
      D         WRITE(IKB,2048)IBBT,BETA,IBETA
      2048      FORMAT(10X,'IBBT-BETA-IBETA:',I3,E15.8,I6)
      CDDDDDDDDDDDDDDDDDDD
      C
      C*****
      C         COMPUTE THE REDUCED WAVEFORM
      C*****
                IF(MPIT .EQ. 0)BETA=0.
                DO 2100 J=1,LTH
                XI(J)=LO(LTH+J)-BETA*LO(LTH+J-MPIT)
      2100      CONTINUE
      C
      C
      C
      C*****
      C         COMPUTE ACF FROM THE REDUCED WAVEFORM
      C*****
      C
                DO 2111 I=1,NNN
                XR(I)=0.0
```

```
0170            ITR=LTH-I+1
0171            DO 2111 J=1,ITR
0172    2111    XR(I)=XR(I)+XI(J)*XI(I+J-1)
        C
        C
        C       NORMALIZE ACF
        C
0173            R0=XR(1)
0174            IF(R0 .LE. 0)R0=1.
0175            DO 2200 I=1,NNN
0176    2200    XR(I)=XR(I)/R0
        C*****
        C       COMPUTE SPECTRAL PREDICTION COEFFICIENTS
        C*****
        C
0177            CALL NR2NAP(ALOW,XR,NLOW,U,PARLOW)
        C
        C
        CDDDDDDDDDDDDDDDDDDDDDDDD
        D       DO 2204 I=1,NLOW
        D       WRITE(IKB,2205)IBPL(I),PARLOW(I),IDPL(I),ALOW(I)
        D2204   CONTINUE
        CDDDDDDDDDDDDDDDDDDDDDDDD
        C
        C       QUANTIZE LOW BAND PARCORS
0178            CALL QTPCRL(PARLOW)
        C
        C
        CDDDDDDDDDDDDDDDDDDDDDDDD
        D       DO 2206 I=1,NLOW
        D       WRITE(IKB,2205)IBPL(I),PARLOW(I),IDPL(I),ALOW(I)
        D2205   FORMAT(20X,'IBPL-PARLOW-IDPL-ALOW:',I3,E15.8,I6,E15.8)
        D2206   CONTINUE
        CDDDDDDDDDDDDDDDDDDDDDDDDD
        C
        C*****
        C       CONVERT PARCORS BACK TO PREDICTOR COEFS
        C*****
0179            CALL PARPRE(NLOW,PARLOW,ALOW)
        C
        C
        CDDDDDDDDDDDDDDDDDDDDDDDDDDD
        D       DO 2207 I=1,NLOW
        D       WRITE(IKB,2208)ALOW(I)
        D2208   FORMAT(30X,E15.8)
        D2207   CONTINUE
        CDDDDDDDDDDDDDDDDDDDDDDDDDDDD
        C
        C*****
        C       CALCULATE GAIN
        C*****
        C
0180            QQL=SQRT(R0*PARLOW(NNN)/FLTH)
0181            IF(QQL .LE. 1.0)QQL=1.0
0182            QQL=ALOG(QQL)/ALOG(2.0)
        C
```

```
                C
                CDDDDDDDDDDDDDDDDDDDDDDD
                D        WRITE(IKB,2220)IBQL,QQL,IQQL
                CDDDDDDDDDDDDDDDDDDDDDDDD
                C
                C        QUANITZE QQL
                C
0183                     IQI=32
0184                     CALL QTZ(IQI,IBQL,QQL,IQQL)
0185                     QQL=2.0**QQL
                C
                C
                CDDDDDDDDDDDDDDDDDDDDDDDDDDDDDD
                D        WRITE(IKB,2220)IBQL,QQL,IQQL
                D2220    FORMAT(10X,'IBQL-QQL-IQQL:',I3,E15.8,I6)
                CDDDDDDDDDDDDDDDDDDDDDDDDDDDDDD
                C
                C
                C        @@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
                C        @                                      @
                C        @      HIGH BAND ENCODING              @
                C        @                                      @
                C        @@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
                C
                C*****
                C        COMPUTE THE SPECTRAL PREDICTION COEFS
                C*****
0186            2300     CONTINUE
0187                     DO 2410 I=1,NNU
0188                     XR(I)=0.
0189                     LTR=LTH-I+1
0190                     DO 2410 J=1,LTR
0191            2410     XR(I)=UP(LTH+J)*UP(LTH+I+J-1)+XR(I)
                C
0192                     R0=XR(1)
0193                     IF(R0.LE.0.)R0=1.
0194                     DO 2420 I=1,NNU
0195            2420     XR(I)=XR(I)/R0
0196                     CALL NR2NAP(AUPP,XR,NUPP,U,PARUPP)
                C
                C
                C
                CDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDD
                D        DO 2421 I=1,NUPP
                D2421    WRITE(IKB,2422)IBPH(I),PARUPP(I),IDPH(I),AUPP(I)
                CDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDD
                C
                C
                C        QUANTIZE HIGH BAND PARCORS
0197                     CALL QTPCRH(PARUPP)
                C
                C
                CDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDD
                D        DO 2424 I=1,NUPP
                D        WRITE(IKB,2422)IBPH(I),PARUPP(I),IDPH(I),AUPP(I)
                D2422    FORMAT(//1X,'IBPH-PARUPP-IDPH-AUPP',I3,E15.8,I6,E15.8)
```

```
          D2424   CONTINUE
          CDDDDDDDDDDDDDDDDDDDDDDDDDDDDDD
          C
          C
          C       CONVERT TO PREDICTOR COEFS
          C
0198              CALL PARPRE(NUPP,PARUPP,AUPP)
          C
          C
          CDDDDDDDDDDDDDDDDDDDDDDDDDDDDD
          D       DO 2425 I=1,NUPP
          D2425   WRITE(IKB,2208)AUPP(I)
          CDDDDDDDDDDDDDDDDDDDDDDDDDDDD
          C
0199              QQH=SQRT(R0*PARUPP(NNU)/FLTH)
0200              IF(QQH .LE.1.0)QQH=1.0
0201              QQH=ALOG(QQH)/ALOG(2.0)
          C
          C
          CDDDDDDDDDDDDDDDDDDDDDDDDD
          D       WRITE(IKB,2430)IBQH,QQH,IQQH
          CDDDDDDDDDDDDDDDDDDDDDDDDD
          C
          C       QUANTIZE QQH
0202              IQI=95
0203              CALL QTZ(IQI,IBQH,QQH,IQQH)
0204              QQH=2.0**QQH
          C
          CDDDDDDDDDDDDDDDDDDDDDDDDDDDD
          D       WRITE(IKB,2430)IBQH,QQH,IQQH
          D2430   FORMAT(//10X,'IBQH-QQH-IQQH',I3,E15.8,I6)
          CDDDDDDDDDDDDDDDDDDDDDDDDDDDD
          C
          C
          C
          C*****
          C       ADAPTIVE BIT ALLOCATIONS
          C*****
          C
          C       FIND IERR AND KERR OR IBIL AND IBIH
          C       ASSUME THE AVERAGE BIT=1.5
0205              FIBIL=1.5+ALOG(QQL/QQH)/ALOG(4.0)
0206              IBIL=FIBIL+0.5
0207              IF(IBIL.GE.ILMX)IBIL=ILMX
0208              IF(IBIL.LE.ILMN)IBIL=ILMN
0209              IBIH=3-IBIL
          C
          C
          C*****
          C       LOW BAND NOISE SHAPING
          C*****
0210              B1L=0.0
0211              B2L=0.0
          C
          C       CALCULATE ACF OF THE LOW-BAND PREDICTION COEFFICIENTS
0212              RL0=1.0
```

```
              RL1=ALOW(1)
              RL2=ALOW(2)
              DO 4522 K=1,NLOW
              K1=K+1
              K2=K+2
              TR=ALOW(K)
              TR1=0.0
              TR2=0.0
              IF(K1.LE.NLOW)TR1=ALOW(K1)
              IF(K2.LE.NLOW)TR2=ALOW(K2)
              RL0=RL0+TR**2
              RL1=RL1+TR*TR1
              RL2=RL2+TR*TR2
      4522    CONTINUE
              DETL=RL0**2-RL1**2
              IF(DETL.GT.1.E-5)GO TO 4524
              QQL=1.E-6
              GOTO 4530
      4524    B1L=RL1*(RL2-RL0)/DETL
              B2L=(RL1**2-RL0*RL2)/DETL
      C
      C*****
      C       HIGH BAND NOISE SHAPING
      C*****
      4530    CONTINUE
      C
              B1H=0.0
              B2H=0.0
      C
      C       CALCULATE ACF OF THE HIGH-BAND PREDICTION COEFFICIENT
              RH0=1.0
              RH1=AUPP(1)
              RH2=AUPP(2)
              DO 4544 K=1,NUPP
              K1=K+1
              K2=K+2
              TR=AUPP(K)
              TR1=0.0
              TR2=0.0
              IF(K1.LE.NUPP)TR1=AUPP(K1)
              IF(K2.LE.NUPP)TR2=AUPP(K2)
              RH0=RH0+TR**2
              RH1=RH1+TR*TR1
              RH2=RH2+TR*TR2
      4544    CONTINUE
              DETH=RH0**2-RH1**2
              IF(DETH.GT.1.E-5)GO TO 4555
              QQL=1.E-6
              QQH=1.E-6
              GOTO 4015
      C
      C
      4555    B1H=RH1*(RH2-RH0)/DETH
              B2H=(RH1**2-RH0*RH2)/DETH
      C
      4015    CONTINUE
```

```
      C
      C*****
      C       CODING THE LOW-BAND:
      C       GENERATE THE TRUE ERROR SIGNAL WITH QUANTIZER IN LOOP
      C*****
      C
      C
      C
              QLEVL=QQL
              DO 2109 J=1,LTH
              SUM=0.
              DO 2108 K=1,NLOW
              TERM=-ALOW(K)*(RNL(LTH+J-K)-BETA*RNL(LTH+J-K-MPIT))
      2108    SUM=SUM+TERM
              SUM=SUM+BETA*RNL(LTH+J-MPIT)
              EL(J)=LO(LTH+J)-SUM+B1L*Q1L+B2L*Q2L
              XIN=EL(J)/QLEVL
              IQI=IQIL
              IF(IBIL.GE.1)CALL ERSQTZ(IQI,IBIL,XIN,XOUT,IDOT)
              NIN(J)=IDOT
              EL(J)=XOUT*QLEVL
              QL=(XOUT-XIN)*QLEVL
              Q2L=Q1L
              Q1L=QL
      2107    RNL(LTH+J)=SUM+EL(J)
      2109    CONTINUE
      C
      C
      C
      C
      C*****
      C       CODING THE HIGH-BAND
      C       GENERATE THE ERROR SIGNAL WITH QUANTIZER IN LOOP
      C*****
              QLEVH=QQH
              DO 3050 J=1,LTH
              SUM=0.
              DO 3040 K=1,NUPP
              TERM=-AUPP(K)*RNH(LTH+J-K)
      3040    SUM=SUM+TERM
              EH(J)=UP(LTH+J)-SUM+B1H*Q1H+B2H*Q2H
              XIN=EH(J)/QLEVH
              IQI=IQIH
              IF(IBIH.GE.1)CALL ERSQTZ(IQI,IBIH,XIN,XOUT,IDOT)
              NIN(J+72)=IDOT
              EH(J)=XOUT*QLEVH
              QH=(XOUT-XIN)*QLEVH
              Q2H=Q1H
              Q1H=QH
      3050    RNH(LTH+J)=EH(J)+SUM

      C
      C
      C       @@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
      C       @                                @
      C       @      FOR DEBUGGING ONLY        @
```

```
          C         @                                   @
          C         @@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
0293                IF(IPRSW.NE.1)GO TO 8444
          C         PRINT TRANSMITTER DATA
          C
0294                WRITE(5,8410)ICOUNT,MPIT,IBIL,IBIH
0295      8410      FORMAT(1X,'FR=',I6,3X,3I6)
0296                WRITE(5,8420)QQL,QQH,BETA,(PARLOW(KK),KK=1,4),(PARUPP(N),N=1,4)
0297                WRITE(5,8430)IQQL,IQQH,IBETA,(IDPL(KK),KK=1,4),(IDPH(N),N=1,4)
0298      8420      FORMAT(10X,11E11.4)
0299      8430      FORMAT(10X,11(I6,5X))
          D         DO 8222 K=1,72
          D         WRITE(5,8333)K,EL(K),EH(K),NIN(K),NIN(K+72)
          D8222     CONTINUE
0300      8333      FORMAT(1X,'EL(',I3,')=',E11.4,3X,E11.4,3X,2I6)
          C
          C
          C
          C
          C*****
          C         SERIALIZATION OF PARAMETERS
          C*****
0301      8444      CONTINUE
0302                CALL SER(INBA,INB,IBIL)
0303                IF(IPRSW.EQ.1)WRITE(5,8450)(INBA(K),K=1,360)
          C
          C
          C++++++++++++++++++++++++++++++++++++++++++++++++
          C
          C         RECEIVER STARTS HERE...
          C
          C++++++++++++++++++++++++++++++++++++++++++++++++
          C*****
          C         BCH CODE ENCODER
          C*****
          C
          C         ENCODE 5 BLOCK
          C
0304                DO 8447 KT=1,5
0305                CALL ENCBCH(INBA,INB,INC,KT)
0306      8447      CONTINUE
          C
          C
          C         PRINT INPUT BINARY DATA IF DESIRED
          C
          C         IF(IPRSW.EQ.1)WRITE(5,8450)(INBA(K),K=1,360)
0307      8450      FORMAT(60I2)
          C
          C
          C
          C         *** END OF TRANSMITTER ***
          C
          C
          C*****
          C         CHANNEL ERROR SIMULATOR
          C*****
```

```
        C
0308                CALL CEIR(INBA,360,CERT,IRN,JRN,NERB,5)
        C
        C           PRINT BINARY DATA IF DESIRED
        C
        C           IF(IPRSW.EQ.1)WRITE(5,8450)(INBA(K),K=1,360)
        C
        C
        C*****
        C           BCH CODE DECODER
        C*****
        C           DECODE 5 BLOCK OF BCH CODE
        C
0309                JSP=0
0310                DO 9700 KT=1,5
0311                CALL DECBCH(INBA,INB,KT,NES)
0312                IF(KT .EQ. 1 .AND. NES.EQ.4)JSP=1
0313    9700        CONTINUE
        C
        C
        C           PRINT INPUT BINARY DATA IF DESIRED
        C
0314                IF(IPRSW.EQ.1)WRITE(5,8450)(INBA(K),K=1,360)
        C
        C           SKIP ONE FRAME SYNTHESIS IF BURST ERRORS(MORE THAN 4 ERRORS) ARE DE
        C           IN THE IST BLOCK(SIDE INFORMATION)
0315                IF(JSP.EQ.1)GO TO 8558
        C
        C*****
        C           DESERIALIZE BINARY DATA INTO DECIMAL NUMBER AND DEQUANTIZE
        C*****
        C
        C
0316                CALL DSER(INBA,INB,QQL,QQH,IBIL,IBIH,ILMX,ILMN)
        C
        D           IF(IPRSW.EQ.1)
        D         1 WRITE(IKB,8430)IBETA,IQQL,IQQH,(IDPL(K),K=1,4),(IDPH(J),J=1,4)
        C
        C           DEQUANTIZE SBAPC PARAMETERS
        C
        C           MPIT DOES NOT REQUIRE DEQUANTIZATION SINCE IBPT=6
        C
        C
        C           DEQUANTIZE BETA
        C
0317                IQI=1
0318                CALL DEQTZ(IQI,IBETA,BETA)
0319                IF(MPIT.EQ.0)BETA=0.0
        D           IF(IPRSW.EQ.1)
        D         1 WRITE(IKB,2048)IBBT,BETA,IBETA
        C
        C           DEQUANTIZE LOWBAND PARCOR COEFS
        C
0320                IQI=158
0321                CALL DQTPCR(IQI,PARLOW)
        C
```

```
0322                IF(IPRSW.EQ.1)WRITE(IKB,8420)(PARLOW(KK),KK=1,4)
          C         CONVERT PARCOR TO FILTER COEFFICIENTS
          C
0323                CALL PARPRE(NLOW,PARLOW,ALOW)
          C
          C         DEQUANTIZE HIGH BAND PARCOR COEFS
          C
0324                IQI=314
0325                CALL DQTPCR(IQI,PARUPP)
          C
          C         CONVERT PARCOR TO FILTER COEFFICIENTS
          C
0326                CALL PARPRE(NUPP,PARUPP,AUPP)
          C
          C         DEQUANTIZE ERROR SIGNALS OF LOW AND HIGH BANDS
          C
0327                CALL ERSDQT(IBIL,EL,QQL,IBIH,EH,QQH)
          C
          C
          C         @@@@@@@@@@@@@@@@@@@@@@
          C         @                    @
          C         @ FOR DEBUGGING ONLY@
          C         @                    @
          C         @@@@@@@@@@@@@@@@@@@@@@
0328                IF(IPRSW.NE.1)GO TO 8555
          C         PRINT RECEIVED DATA
          C
0329                WRITE(5,8410)ICOUNT,MPIT,IBIL,IBIH
0330                WRITE(5,8420)QQL,QQH,BETA,(PARLOW(KK),KK=1,4),(PARUPP(N),N=1,4
0331                WRITE(5,8430)IQQL,IQQH,IBETA,(IDPL(KK),KK=1,4),(IDPH(N),N=1,4)
          D         DO 8522 K=1,72
          D         WRITE(5,8333)K,EL(K),EH(K),NIN(K),NIN(K+72)
          D8522     CONTINUE
0332      8555      CONTINUE
          C
          C
          C****
          C         SYNTHESIZE LOW BAND OUTPUTS
          C****
0333                DO 3001 J=1,LTH
0334                SUM=0.
0335                DO 3000 K=1,NLOW
0336                TERM=-ALOW(K)*(CLOW(LTH+J-K)-BETA*CLOW(LTH+J-K-MPIT))
0337      3000      SUM=SUM+TERM
0338      3001      CLOW(LTH+J)=EL(J)+SUM+BETA*CLOW(LTH+J-MPIT)
          C
          C         WRITE(IKB,3100)(CLOW(LTH+J),J=1,LTH)
          C3100     FORMAT(1X,E12.5)
          C
          C
          C****
          C         SYNTHESIZE HIGH BAND OUTPUTS
          C****
0339                DO 3301 J=1,LTH
0340                SUM=0.
0341                DO 3300 K=1,NUPP
```

```
                TERM=-AUPP(K)*CUPP(LTH+J-K)
      3300      SUM=SUM+TERM
      3301      CUPP(LTH+J)=EH(J)+SUM
      C
      C         WRITE(IKB,3100)(CUPP(LTH+J),J=1,LTH)
      C
      C
      C
      C
      C
      C
      C
      C         ZERO OUT BUFFERS AND INSERT DOWN-SAMPLED VALUES
      C
      8558      DO 4001 I=1,LX,2
                YYL(I+LX)=0.
      4001      YYH(I+LX)=0.
      C
      C
                DO 4010 I=1,LTH
                YYL(LX+2*I)=2.*CLOW(I+LTH)
      4010      YYH(LX+2*I)=2.*CUPP(I+LTH)
      C
      C
      C*****
      C         QMF FILTERING
      C*****
                DO 4030 J=1,LX
                SUM=0.
                SUM1=0.
                DO 4020 I=1,NFILT
                SUM1=SUM1+HUPP(I)*YYH(LX+J-I+1)
      4020      SUM=SUM+HLOW(I)*YYL(LX+J-I+1)
                TT=SUM-SUM1
                IF(TT.GE.2047.)TT=2047.
                IF(TT.LE.-2047.)TT=-2047.
                XR(J)=TT
      4030      NOUT(J)=TT
      C
      C         ++++++++++ END  RECEIVER ++++++++++
      C
      C
      C
      C
      C
      C****
      C         WRITE OUT THE DATA
      C*****
                CALL TAPE2(2)
      C
      C
      C****
      C         COMPUTE SIGNAL-TO-NOISE RATIOS
      C****
                IF(ISNR  .EQ.1)GOTO 4402
                KOUNT=KOUNT+1
```

```
              SNRNUM=0.
              SNRDEN=0.
              DO 4100 I=1,LX
              SNRNUM=SNRNUM+(XBUF(I+LX+1-NFILT))**2
     4100     SNRDEN=SNRDEN+(XBUF(I+LX+1-NFILT)-XR(I))**2
              IF(SNRDEN .EQ. 0 .OR.SNRNUM .EQ. 0.)GOTO 4115
              SSNR=10.*ALOG10(SNRNUM/SNRDEN)
              GOTO 4120
     4115     SSNR=0.
              WRITE(IKB,4117)ICOUNT
     4117     FORMAT(1X,'FRAME NO.=',I4,'ENERGY OF INPUT SIGNAL=0.')
     4120     XSNR=XSNR+SSNR
              CSNR=XSNR/FLOAT(KOUNT)
              WRITE(IKB,4122)ICOUNT,SSNR,CSNR,(NERB(L),L=1,6)
     4122     FORMAT(1X,'FR #=',I4,4X,'SNR=',E11.4,'DB',
         1    3X,'CSNR=',E11.4,3X,'CH. ERRS',6I3)
     C
     C
     C
     C*****
     C        THIS SHIFTS THE DATA FOR NEXT TIME
     C*****
     C
     4402     CONTINUE
              DO 4403 I=1,LX
              XBUF(I)=XBUF(I+LX)
              YYL(I)=YYL(I+LX)
     4403     YYH(I)=YYH(I+LX)
     C
     C
              DO 4404 J=1,LTH
              LO(J)=LO(LTH+J)
              UP(J)=UP(LTH+J)
              RNL(J)=RNL(LTH+J)
              RNH(J)=RNH(J+LTH)
              CLOW(J)=CLOW(J+LTH)
              CUPP(J)=CUPP(J+LTH)
     4404     CONTINUE
     C
     C***
     C        CONTINUE PROCESSING IF END OF TAPE IS NOT READ
     C***
              IF(NEND .EQ. 0)GOTO 1000
     C
              GOTO 5000
     C
     C
     C****
     C        ERROR DIAGNOSTICS
     C****
     4900     WRITE(IKB,4910)
     4910     FORMAT(1X,'ERROR ENCOUNTERED DURING TAPE READ'//)
              GOTO 70
     C
     C
     C
```

```
C*****
C       WRITES THE END OF FILE
C*****
5000    CONTINUE
        DO 5001 I=1,LX
5001    NOUT(I)=0
        DO 5002 I=1,32
5002    CALL TAPE2(2)
        CALL TAPE2(5)
        WRITE(IKB,5109)
5109    FORMAT(1X,'MISSION ACCOMPLISHED'/)
        STOP
        END
```

PROGRAM SECTIONS

NAME	SIZE		ATTRIBUTES
$CODE1	011762	2553	RW,I,CON,LCL
$PDATA	000056	23	RW,D,CON,LCL
$IDATA	001334	366	RW,D,CON,LCL
$VARS	025664	5594	RW,D,CON,LCL
$TEMPS	000006	3	RW,D,CON,LCL
MTAPE0	001250	340	RW,D,OVR,GBL
MTAPE1	000012	5	RW,D,OVR,GBL
MTAPE2	000012	5	RW,D,OVR,GBL
TBL	003444	914	RW,D,OVR,GBL
NSTBL	000310	100	RW,D,OVR,GBL
SBTA	000030	12	RW,D,OVR,GBL
SDTA	000030	12	RW,D,OVR,GBL
SW	000004	2	RW,D,OVR,GBL

TOTAL SPACE ALLOCATED = 046622 9929

```
      C
      C
      C
      C
      C         @@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
      C         @                              @
      C         @ SUBROUTINES START HERE       @
      C         @                              @
      C         @@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
      C         SUBROUTINE NR2NAP.FTN(MODIFIED FROM SOLVE.FTN)
      C         THIS ROUTINE CONVERTS NORMALIZED AUTOCORRELATION COEFS
      C         TO NEGATIVE PREDICTIVE COEFS &PARCOR COEFS
      C         THE RESULTING A'S SHOULD BE NEGATED IF
      C         (1-SUM(A(I)*Z**(-I)))IS USED AS THE PREDICTION POLYNOMIAL
      C
      C
      C
                SUBROUTINE NR2NAP(A,R,N,U,PARCOR)
                DIMENSION R(1),PARCOR(1),A(1),B(10)
                A(1)=-R(2)
                PARCOR(1)=-A(1)
                U=1.0+A(1)*R(2)
                DO 3 I=2,N
                W=R(I+1)
                L=I-1
                DO 1 M=1,L
                B(M)=A(I-M)
      1         W=W+B(M)*R(M+1)
                XK=-W/U
                L=I-1
                DO 2 M=1,L
      2         A(M)=A(M)+XK*B(M)
                A(I)=XK
                PARCOR(I)=-XK
      3         U=U+XK*W
                RETURN
                END
```

PROGRAM SECTIONS

NAME	SIZE		ATTRIBUTES
$CODE1	000452	149	RW,I,CON,LCL
$IDATA	000036	15	RW,D,CON,LCL
$VARS	000066	27	RW,D,CON,LCL
$TEMPS	000006	3	RW,D,CON,LCL

TOTAL SPACE ALLOCATED = 000604   194

```
          C
          C
          C          SUBROUTINE PARPRE.FTN
          C          THIS ROUTINE CONVERTS PARCOR COEFS TO PREDICTION COEFS
          C          THE RESULTING A'S SHOULD BE NEGATED IF
          C          (1-SUM(A(I)*Z**(-I)) IS USED AS THE PREDICTION POLYNOMIAL
          C
0001                 SUBROUTINE PARPRE(N,PARCOR,A)
0002                 DIMENSION PARCOR(1),A(1),AP(10)
0003                 A(1)=-PARCOR(1)
0004                 NN=N+1
0005                 PARCOR(NN)=1.0-A(1)**2
0006                 DO 120 I=2,N
0007                 IM1=I-1
0008                 DO 110 J=1,IM1
0009      110        AP(J)=A(J)-PARCOR(I)*A(I-J)
0010                 AP(I)=-PARCOR(I)
0011                 PARCOR(NN)=PARCOR(NN)*(1.0-AP(I)**2)
0012                 DO 140 J=1,I
0013      140        A(J)=AP(J)
0014      120        CONTINUE
          C          TYPE 997,(I,A(I),I=1,N)
          C997       FORMAT(1X,'A(',I2,')= ',E15.8)
0015                 RETURN
0016                 END
```

PROGRAM SECTIONS

NAME	SIZE		ATTRIBUTES
$CODE1	000372	125	RW,I,CON,LCL
$IDATA	000024	10	RW,D,CON,LCL
$VARS	000060	24	RW,D,CON,LCL
$TEMPS	000004	2	RW,D,CON,LCL

TOTAL SPACE ALLOCATED = 000502 161

```
C
C
C        SUBROUTINE QTPCRL
C        QUANTIZE PARAMETERS OF LOW BAND PARCORS IN SBAPC
C
         SUBROUTINE QTPCRL(PCRL)
         COMMON/TBL/QTBL(457)
         COMMON/SBTA/IBPT,IBBT,IBQL,IBQH,IBPL(4),IBPH(4)
         COMMON/SDTA/MPIT,IBETA,IQQL,IQQH,IDPL(4),IDPH(4)
         DIMENSION PCRL(1)
C
C        PCRL(1,2) WITH 5 BITS
C        PCRH(1,2) WITH 4 BITS
C        PCRL(3,4),PCRH(3,4) WITH 3 BITS
C
C        QUANTIZE PARCOR IN LOW BAND
         IQS=158
         IQI=IQS
         DO 110 I=1,4
         IBT=IBPL(I)
         CALL QTZ(IQI,IBT,PCRL(I),IDP)
         IDPL(I)=IDP
         IQS=IQS+15
         IF(IBPL(I).GE.4)IQS=IQS+16
         IF(IBPL(I).GE.5)IQS=IQS+32
         IQI=IQS
  110    CONTINUE
         RETURN
         END
```

PROGRAM SECTIONS

NAME	SIZE		ATTRIBUTES
SCODE1	000216	71	RW,I,CON,LCL
SIDATA	000024	10	RW,D,CON,LCL
SVARS	000012	5	RW,D,CON,LCL
STEMPS	000002	1	RW,D,CON,LCL
TBL	003444	914	RW,D,OVR,GBL
SBTA	000030	12	RW,D,OVR,GBL
SDTA	000030	12	RW,D,OVR,GBL

TOTAL SPACE ALLOCATED = 004002 1025

NO FPP INSTRUCTIONS GENERATED

```
C
C
C       SUBROUTINE QTPCRH
C       SUBROUTINE TO QUANTIZE HIGH BAND PARCOR IN SBAPC
C
        SUBROUTINE QTPCRH(PCRH)
        COMMON/TBL/QTBL(457)
        COMMON/SBTA/IBPT,IBBT,IBQL,IBQH,IBPL(4),IBPH(4)
        COMMON/SDTA/MPIT,IBETA,IQQL,IQQH,IDPL(4),IDPH(4)
        DIMENSION PCRH(1)
C
C       QUANTIZE PARCOR IN UPPER BAND
C
        IQS=314
        IQI=IQS
        DO 210 I=1,4
        IBT=IBPH(I)
        CALL QTZ(IQI,IBT,PCRH(I),IDP)
        IDPH(I)=IDP
        IQS=IQS+15
        IF(IBPH(I).GE.4)IQS=IQS+16
        IQI=IQS
  210   CONTINUE
        RETURN
        END
```

PROGRAM SECTIONS

NAME	SIZE		ATTRIBUTES
$CODE1	000202	65	RW,I,CON,LCL
$IDATA	000024	10	RW,D,CON,LCL
$VARS	000012	5	RW,D,CON,LCL
$TEMPS	000002	1	RW,D,CON,LCL
TBL	003444	914	RW,D,OVR,GBL
SBTA	000030	12	RW,D,OVR,GBL
SDTA	000030	12	RW,D,OVR,GBL

TOTAL SPACE ALLOCATED = 003766 1019

NO FPP INSTRUCTIONS GENERATED

```
C       QTZ.FTN
C       SUBROUTINE  TO QUANTIZE SIDE INFORMATION OF SBAPC SYSTEM
C
        SUBROUTINE QTZ(IQI,IBT,XX,ID)
        COMMON/TBL/QTBL(457)
        ID=0
        IF(IBT.LE.0)GO TO 20
        IQIT=2**IBT-1
        DO 10 J=1,IQIT
        ID=J-1
        IF(XX.LT.QTBL(IQI+1))GO TO 20
        IQI=IQI+2
10      CONTINUE
        ID=ID+1
20      XX=QTBL(IQI)
        RETURN
        END
```

PROGRAM SECTIONS

NAME	SIZE		ATTRIBUTES
$CODE1	000154	54	RW,I,CON,LCL
$VARS	000004	2	RW,D,CON,LCL
TBL	003444	914	RW,D,OVR,GBL

TOTAL SPACE ALLOCATED = 003624 970

```
      C       DEQTZ.FTN
      C
      C       DEQUANTIZE SBAPC PARAMETERS
0001          SUBROUTINE DEQTZ(IQI,ID,XX)
0002          COMMON/TBL/QTBL(457)
      C       IQI:INPUT TABLE POINTER
      C       ID=DECIMAL INPUT VALUE
      C       XX:DEQUANTIZED OUTPUT VALUE
0003          IQ=IQI+2*ID
0004          XX=QTBL(IQ)
0005          RETURN
0006          END
```

PROGRAM SECTIONS

NAME	SIZE		ATTRIBUTES
$CODE1	000036	15	RW,I,CON,LCL
$VARS	000002	1	RW,D,CON,LCL
TBL	003444	914	RW,D,OVR,GBL

TOTAL SPACE ALLOCATED = 003504 930

```
C       ERSQTZ.FTN
C       SUBROUTINE TO QUANTIZE ERROR SIGNALS
C       DATE 7/31/80
        SUBROUTINE ERSQTZ(IQI,IBIT,XIN,XOUT,IDOT)
        COMMON/TBL/QTBL(457)
        IF(IBIT.GE.2)IQI=IQI+1
        IF(IBIT.GE.3)IQI=IQI+3
        IF(IBIT.GE.4)IQI=IQI+7
        XOUT=ABS(XIN)
        IBT=IBIT-1
        IMK=2**IBT
        CALL QTZ(IQI,IBT,XOUT,IDOT)
        IF(XIN.GE.0.0)RETURN
        XOUT=-XOUT
        IDOT=IDOT+IMK
        RETURN
        END
```

PROGRAM SECTIONS

NAME	SIZE		ATTRIBUTES
$CODE1	000200	64	RW,I,CON,LCL
$IDATA	000012	5	RW,D,CON,LCL
$VARS	000004	2	RW,D,CON,LCL
TBL	003444	914	RW,D,OVR,GBL

TOTAL SPACE ALLOCATED = 003662 985

```
C       DQTPCR.FTN
C       SUBROUTINE DEQUANTIZE PARCOR PARAMETERS
C
C       AUG. 4, 1980
C
        SUBROUTINE DQTPCR(IQIP,PCR)
        COMMON/TBL/QTBL(457)
        COMMON/SBTA/IBPT,IBBT,IBQL,IBQH,IBPL(4),IBPH(4)
        COMMON/SDTA/MPIT,IBETA,IQQL,IQQH,IDPL(4),IDPH(4)
        DIMENSION PCR(1)
        IQI=IQIP
        DO 10 I=1,4
        IF(IQIP.EQ.314)GO TO 11
        IDP=IDPL(I)
        CALL DEQTZ(IQI,IDP,PCR(I))
        IBIT=IBPL(I)
        GO TO 12
C       DEQUANTIZE FOR HIGH BAND
11      CONTINUE
        IDP=IDPH(I)
        CALL DEQTZ(IQI,IDP,PCR(I))
        IBIT=IBPH(I)
12      CONTINUE
        IQI=IQI+15
        IF(IBIT.GE.4)IQI=IQI+16
        IF(IBIT.GE.5)IQI=IQI+32
10      CONTINUE
        RETURN
        END
```

PROGRAM SECTIONS

NAME	SIZE		ATTRIBUTES
$CODE1	000304	98	RW,I,CON,LCL
$IDATA	000022	9	RW,D,CON,LCL
$VARS	000010	4	RW,D,CON,LCL
$TEMPS	000002	1	RW,D,CON,LCL
TBL	003444	914	RW,D,OVR,GBL
SBTA	000030	12	RW,D,OVR,GBL
SDTA	000030	12	RW,D,OVR,GBL

TOTAL SPACE ALLOCATED = 004064 1050

0 FPP INSTRUCTIONS GENERATED

```
      C       ERSDQT.FTN
      C
      C       ERROR SIGNALS DEQUANTIZER
      C       AUG. 4, 1980
      C
              SUBROUTINE ERSDQT(IBIL,EL,QQL,IBIH,EH,QQH)
              COMMON/MTAPE0/NIN(170),NOUT(170)
              COMMON/TBL/QTBL(457)
              COMMON/SBTA/IBPT,IBBT,IBQL,IBQH,IBPL(4),IBPH(4)
              COMMON/SDTA/MPIT,IBETA,IQQL,IQQH,IDPL(4),IDPH(4)
              DIMENSION EL(1),EH(1)
              DATA SQRT3,IRN,JRN/1.73201,0,0/
              IQL=406
              IF(IBIL.GE.2)IQL=IQL+1
              IF(IBIL.GE.3)IQL=IQL+3
              IF(IBIL.GE.4)IQL=IQL+7
              ISGL=2**(IBIL-1)
              IQH=432
              IF(IBIH.GE.2)IQH=IQH+1
              IF(IBIH.GE.3)IQH=IQH+3
              IF(IBIH.GE.4)IQH=IQH+7
              ISGH=2**(IBIH-1)
              DO 10 I=1,72
      C
      C       DEQUANTIZE FOR LOW BAND
              IDL=NIN(I)
              IF(IDL.GE.ISGL)IDL=IDL-ISGL
              IF(IBIL.EQ.0)GO TO 20
              CALL DEQTZ(IQL,IDL,EL(I))
              EL(I)=EL(I)*QQL
              IF(NIN(I).GE.ISGL)EL(I)=-EL(I)
              GO TO 30
      20      CONTINUE
      C
      C       ADD RANDOM NUMBER FOR ERROR SIGNALS
      C
              CALL RANDU(IRN,JRN,YQR)
              YQR=(YQR+YQR-1.0)*SQRT3
              EL(I)=YQR*SQRT3
      30      CONTINUE
              IDH=NIN(I+72)
              IF(IDH.GE.ISGH)IDH=IDH-ISGH
              IF(IBIH.EQ.0)GO TO 40
              CALL DEQTZ(IQH,IDH,EH(I))
              EH(I)=EH(I)*QQH
              IF(NIN(I+72).GE.ISGH)EH(I)=-EH(I)
              GO TO 10
      40      CONTINUE
              CALL RANDU(IRN,JRN,YQR)
              YQR=(YQR+YQR-1.0)*SQRT3
              EH(I)=YQR*SQRT3
      10      CONTINUE
              RETURN
              END
```

```
PROGRAM SECTIONS

 NAME        SIZE                    ATTRIBUTES

SCODE1  000744   242                 RW,I,CON,LCL
SIDATA  000054    22                 RW,D,CON,LCL
SVARS   000032    13                 RW,D,CON,LCL
STEMPS  000004     2                 RW,D,CON,LCL
MTAPEO  001250   340                 RW,D,OVR,GBL
TBL     003444   914                 RW,D,OVR,GBL
SBTA    000030    12                 RW,D,OVR,GBL
SDTA    000030    12                 RW,D,OVR,GBL


TOTAL SPACE ALLOCATED = 006052  1557

SBAPC,LP:=SBAPC/NOTR
```

```
          C       TAPE2.FTN
0001              SUBROUTINE TAPE2(IQ)
0002              IMPLICIT INTEGER(A-Z)
0003              COMMON/MTAPE0/NIN(170),NOUT(170)
0004              COMMON/MTAPE1/NSKIP,IST,NTOTI,NTUPS,NTOTO
0005              COMMON/MTAPE2/NEND,NERR,NFILE,NINS,NOUTS
0006              COMMON/MTAPE3/NBF(1324),NBUF(1324)
0007              COMMON/MTAPE4/LST,IBEG
0008              DIMENSION ICARD(64)
0009            GO TO (700,800,900,999,1000,1001,1002,1003),IQ
          C
          C     INPUT DATA
0010      700     IST=IST+NTUPS
0011              IF(1324.GE.IST) GO TO 200
0012        100 IST=IST-1024
0013            NSKIP=NSKIP+1
0014      200   KOVER=LST+NTOTI-1325
0015              IF(KOVER.GT.0) GO TO 305
0016              DO 5000 I=1,NTOTI
0017      5000    NIN(I)=NBF(LST+I-1)
0018            LST=LST+NTUPS
0019              RETURN
0020        305 IPEP=LST-1024
0021              DO 5001 I=1,300
0022      5001    NBF(IPEP+I-1)=NBF(LST+I-1)
0023            LST=IPEP
0024              IF(NINS.EQ.0) GO TO 2100
          C       DISK INPUT
0025              DO 5010 I=1,16
0026              READ(2,END=2001,ERR=6000)(ICARD(J),J=1,64)
0027              K=64*(I-1)+300
0028              DO 5010 J=1,64
0029      5010    NBF(K+J)=ICARD(J)
0030              IF(NERR.NE.0) GO TO 6000
0031              GO TO 200
0032      2001    NEND=1
0033              GO TO 200
0034      2100    CALL TIN(NBF(301),NEND,NERR)
0035            GO TO 200
          C
0036        800 CONTINUE
          C     OUTPUT DATA
0037              DO 5005 I=1,NTOTO
0038      5005    NBUF(IBEG+I-1)=NOUT(I)
0039              IBEG=IBEG+NTOTO
0040              KON=1025-IBEG
0041              IF(KON.GT.0) GO TO 90
0042              IF(NOUTS.EQ.0) GO TO 2002
0043              DO 6010 I=1,16
0044              K=64*(I-1)
0045              DO 6011 J=1,64
0046      6011    ICARD(J)=NBUF(K+J)
0047      6010    WRITE(3)(ICARD(J),J=1,64)
0048              GO TO 2003
0049      2002    CALL TOUT(NBUF,NERR)
0050      2003    LRESID=-KON
```

```
              DO 5006 I=1,LRESID
      5006    NBUF(I)=NBUF(1024+I)
            IBEG=LRESID+1
      90      RETURN
      C
      C     INITIALIZE
      900     IF((NINS+NOUTS).LE.1) CALL ATTACH
              IF(NINS.EQ.0) CALL RWND0
              IF(NOUTS.EQ.0) CALL RWND1
              IF(NFILE.EQ.0) GO TO 995
              IF(NINS.EQ.0) CALL FSRCH(NFILE,NFRR)
              IF(NERR.NE.0) GO TO 6000
      913     DO 912 J=1,NSKIP
              IF(NINS.EQ.0) GO TO 3000
      C       DISK INPUT
              DO 5011 I=1,16
              READ(2,END=3001,ERR=6000)(ICARD(JJ),JJ=1,64)
              K=64*(I-1)+300
              DO 5011 JJ=1,64
      5011    NBF(K+JJ)=ICARD(JJ)
              GO TO 912
      3001    NEND=1
              GO TO 912
      3000    CALL TIN(NBF(301),NEND,NERR)
              IF(NERR.NE.0) GO TO 6000
        912 CONTINUE
            GO TO 995
        999 CONTINUE
              IBEG=1
              CALL EOFSH(NERR)
              RETURN
        995 CONTINUE
            IBEG=1
            LST=IST+300
            IST=IST-NTUPS
            RETURN
      C     END OF FILE
      1000    IF(NOUTS.EQ.0) GO TO 2010
              CALL CLOSE(3)
              GO TO 2011
      2010    CALL EOFW(NERR)
      2011    IBEG=1
              RETURN
      1001    NEND=0
              IF(NINS.EQ.0) GO TO 4020
              REWIND 2
              GO TO 4011
      4020    CALL RWND0
              NERR=0
              CALL FSRCH(NFILE,NERR)
              IF(NERR.NE.0) GO TO 6000
      4011    DO 950 J=1,NSKIP
              IF(NINS.EQ.0) GO TO 4000
      C       DISK INPUT
              DO 5012 I=1,16
              READ(2,END=4001,ERR=6000)(ICARD(K),K=1,64)
```

```
               K=64*(I-1)+300
               DO 5012 JJ=1,64
      5012     NBF(K+JJ)=ICARD(JJ)
               GO TO 950
      4001     NEND=1
               GO TO 4002
      4000     CALL TIN(NBF(301),NEND,NERR)
               IF(NERR.NE.0) GO TO 6000
      4002     IF(NEND.NE.0) GO TO 2000
      950      CONTINUE
               LST=IST+300
               IST=IST-NTUPS
               RETURN
      2000     NERR=16384
               RETURN
      1002     IF(NINS.EQ.1)CALL CLOSE(2)
               NEND=0
               RETURN
      1003     CALL INFO
               RETURN
      6000     TYPE 6001
      6001     FORMAT(1X,'INPUT FILE ERROR'/)
               NERR=1
               RETURN
             END
```

PROGRAM SECTIONS

NAME	SIZE		ATTRIBUTES
$CODE1	002400	640	RW,I,CON,LCL
$PDATA	000032	13	RW,D,CON,LCL
$IDATA	000070	28	RW,D,CON,LCL
$VARS	000220	72	RW,D,CON,LCL
$TEMPS	000004	2	RW,D,CON,LCL
MTAPE0	001250	340	RW,D,OVR,GBL
MTAPE1	000012	5	RW,D,OVR,GBL
MTAPE2	000012	5	RW,D,OVR,GBL
MTAPE3	012260	2648	RW,D,OVR,GBL
MTAPE4	000004	2	RW,D,OVR,GBL

TOTAL SPACE ALLOCATED = 016526 3755

NO FPP INSTRUCTIONS GENERATED

```
C       PROGRAM FSIO.FTN TO MOVE MAG TAPES
C       AND WRITE SPEECH FOR REAL-TIME I/O USING QIO
C
        SUBROUTINE ATTACH
        IMPLICIT INTEGER(A-Z)
        COMMON/MTAPE2/NEND,NERR,NFILE,NINS,NOUTS
        COMMON/MTAPE5/MASK,ISW(2),IOATT,IOSUC,IEALN,IORWD,
     1  IOWLB,IEVER,IOSPF,IEEOF,IOEOF,IORLB,MT0(6),MT1(6),DSW
        DATA IOATT,IOSUC,IEALN/0001400,1,-34/
        DATA IORWD,IOWLB,IEVER,IOSPF,IORLB/02400,0400,-4,02440,0100
        DATA IOSPF,IEEOF,IOEOF/02440,-10,03000/
        DATA MASK/0377/
        DATA MT0/0,2048,0,0,0,0/
        DATA MT1/0,2048,0,0,0,0/
        IF(NINS.NE.0) GO TO 1
        CALL ASNLUN(2,'MT',0,DSW)
        IF(DSW.EQ.1)GO TO 10
11      WRITE(5,100)
100     FORMAT(1X,'MT0: ATTACH UNSUCCESSFUL'/)
        NERR=1
        RETURN
10      CALL WTQIO(IOATT,2,1,0,ISW,0,DSW)
        IF(IOSUC.EQ.IAND(MASK,ISW(1)))GO TO 1
        IF(IAND(IEALN,MASK).NE.IAND(MASK,ISW(1)))GO TO 11
1       IF(NOUTS.NE.0) GO TO 2
        CALL ASNLUN(3,'MT',1,DSW)
        IF(DSW.EQ.1) GO TO 20
12      WRITE(5,101)
101     FORMAT(1X,'MT1: ATTACH UNSUCCESSFUL'/)
        NERR=1
        RETURN
20      CALL WTQIO(IOATT,3,1,0,ISW,0,DSW)
        IF(IOSUC.EQ.IAND(MASK,ISW(1)))GO TO 2
        IF(IAND(IEALN,MASK).NE.IAND(MASK,ISW(1)))GO TO 12
2       RETURN
        END
```

PROGRAM SECTIONS

NAME	SIZE		ATTRIBUTES
$CODE1	000262	89	RW,I,CON,LCL
$PDATA	000024	10	RW,D,CON,LCL
$IDATA	000160	56	RW,D,CON,LCL
MTAPE2	000012	5	RW,D,OVR,GBL
MTAPE5	000064	26	RW,D,OVR,GBL

TOTAL SPACE ALLOCATED = 000564 186

NO FPP INSTRUCTIONS GENERATED

```
      C
              SUBROUTINE RWNDO
              IMPLICIT INTEGER(A-Z)
              COMMON/MTAPE2/NEND,NERR,NFILE,NINS,NOUTS
              COMMON/MTAPE5/MASK,ISW(2),IOATT,IOSUC,IEALN,IORWD,
          1   IOWLB,IEVER,IOSPF,IEEOF,IOEOF,IORLB,MT0(6),MT1(6),DSW
              CALL WTQIO(IORWD,2,1,0,ISW,0,DSW)
              IF(IOSUC.EQ.IAND(MASK,ISW(1)))GO TO 1
              WRITE(5,902)
      902     FORMAT(1X,'MT0: BUSY'/)
              NERR=1
      1       RETURN
              END
```

PROGRAM SECTIONS

NAME	SIZE		ATTRIBUTES
$CODE1	000062	25	RW,I,CON,LCL
$PDATA	000014	6	RW,D,CON,LCL
$IDATA	000036	15	RW,D,CON,LCL
MTAPE2	000012	5	RW,D,OVR,GBL
MTAPE5	000064	26	RW,D,OVR,GBL

TOTAL SPACE ALLOCATED = 000232 77

NO FPP INSTRUCTIONS GENERATED

```
FORTRAN IV-PLUS V02-51E          16:39:20    06-OCT-80            PAGE 6
TAPE2.FTN        /WR

        C
                      SUBROUTINE RWND1
                      IMPLICIT INTEGER(A-Z)
                      COMMON/MTAPE2/NEND,NERR,NFILE,NINS,NOUTS
                      COMMON/MTAPE5/MASK,ISW(2),IOATT,IOSUC,IEALN,IORWD,
                    1 IOWLB,IEVER,IOSPF,IEEOF,IOEOF,IORLB,MT0(6),MT1(6),DSW
                      CALL WTQIO(IORWD,3,1,0,ISW,0,DSW)
                      IF(IOSUC.EQ.IAND(MASK,ISW(1)))GO TO 1
                      WRITE(5,902)
                      NERR=1
        902           FORMAT(1X,'MT1: BUSY'/)
        1             RETURN
                      END

PROGRAM SECTIONS

 NAME       SIZE                   ATTRIBUTES

$CODE1  000062    25               RW,I,CON,LCL
$PDATA  000014     6               RW,D,CON,LCL
$IDATA  000036    15               RW,D,CON,LCL
MTAPE2  000012     5               RW,D,OVR,GBL
MTAPE5  000064    26               RW,D,OVR,GBL

TOTAL SPACE ALLOCATED = 000232    77

NO FPP INSTRUCTIONS GENERATED
```

```
C
      SUBROUTINE TIN(BUF,NEND,NERR)
      IMPLICIT INTEGER(A-Z)
      COMMON/MTAPE5/MASK,ISW(2),IOATT,IOSUC,IEALN,IORWD,
     1 IOWLB,IEVER,IOSPF,IEEOF,IOEOF,IORLB,MT0(6),MT1(6),DSW
      NEND=0
      NERR=0
      CALL GETADR(MT0,BUF)
      CALL WTQIO(IORLB,2,1,0,ISW,MT0,DSW)
      IF(IOSUC.EQ.IAND(MASK,ISW(1)))GO TO 1
      IF(IAND(IEEOF,MASK).EQ.IAND(MASK,ISW(1)))NEND=1
      IF(IAND(IEVER,MASK).EQ.IAND(MASK,ISW(1)))NERR=1
    1 RETURN
      END
```

PROGRAM SECTIONS

NAME	SIZE		ATTRIBUTES
$CODE1	000156	55	RW,I,CON,LCL
$PDATA	000014	6	RW,D,CON,LCL
$IDATA	000026	11	RW,D,CON,LCL
MTAPE5	000064	26	RW,D,OVR,GBL

TOTAL SPACE ALLOCATED = 000304 98

NO FPP INSTRUCTIONS GENERATED

```
C
            SUBROUTINE TOUT(NBUF,NERR)
            IMPLICIT INTEGER(A-Z)
            COMMON/MTAPE5/MASK,ISW(2),IOATT,IOSUC,IEALN,IORWD,
     1      IOWLB,IEVER,IOSPF,IEEOF,IOEOF,IORLB,MT0(6),MT1(6),DSW
            NERR=0
            CALL GETADR(MT1,NBUF)
            CALL WTQIO(IOWLB,3,1,0,ISW,MT1,DSW)
            IF(IOSUC.EQ.IAND(MASK,ISW(1)))GO TO 1
            IF(IAND(IEVER,MASK).EQ.IAND(MASK,ISW(1)))NERR=1
      1     RETURN
            END
```

PROGRAM SECTIONS

NAME	SIZE		ATTRIBUTES
$CODE1	000110	36	RW,I,CON,LCL
$PDATA	000014	6	RW,D,CON,LCL
$IDATA	000026	11	RW,D,CON,LCL
MTAPE5	000064	26	RW,D,OVR,GBL

TOTAL SPACE ALLOCATED = 000236 79

NO FPP INSTRUCTIONS GENERATED

```
C
      SUBROUTINE FSRCH(NFILE,NERR)
      IMPLICIT INTEGER (A-Z)
      COMMON/MTAPE3/NBF(1324),NBUF(1324)
      COMMON/MTAPE5/MASK,ISW(2),IOATT,IOSUC,IEALN,IORWD,
     1 IOWLB,IEVER,IOSPF,IEEOF,IOEOF,IORLB,MT0(6),MT1(6),DSW
      NERR=0
      FILE=NFILE-1
      IF(FILE.LE.0) RETURN
      DO 1 I=1,FILE
      CALL GETADR(MT0,NBF(301))
      CALL WTQIO(IORLB,2,1,0,ISW,MT0,DSW)
      IF(IOSUC.EQ.IAND(MASK,ISW(1)))GOTO 2
      WRITE(5,100)NFILE
100   FORMAT(1X,'FILE',I4, ' NOT FOUND'/)
      NERR=1
      RETURN
2     MT0(1)=1
1     CALL WTQIO(IOSPF,2,1,0,ISW,MT0,DSW)
      RETURN
      END
```

PROGRAM SECTIONS

NAME	SIZE		ATTRIBUTES
SCODE1	000202	65	RW,I,CON,LCL
SPDATA	000014	6	RW,D,CON,LCL
SIDATA	000076	31	RW,D,CON,LCL
SVARS	000004	2	RW,D,CON,LCL
STEMPS	000002	1	RW,D,CON,LCL
MTAPE3	012260	2648	RW,D,OVR,GBL
MTAPE5	000064	26	RW,D,OVR,GBL

TOTAL SPACE ALLOCATED = 012666 2779

NO FPP INSTRUCTIONS GENERATED

```
      C
            SUBROUTINE EOFSH(NERR)
            IMPLICIT INTEGER(A-Z)
            COMMON/MTAPE3/NBF(1324),NBUF(1324)
            COMMON/MTAPE5/MASK,ISW(2),IOATT,IOSUC,IEALN,IORWD,
          1 IOWLB,IEVER,IOSPF,IEEOF,IOEOF,IORLB,MT0(6),MT1(6),DSW
            NERR=0
      1     CALL GETADR(MT1(1),NBUF(1))
            CALL WTQIO(IORLB,3,1,0,ISW,MT1,DSW)
            IF(IOSUC.EQ.IAND(MASK,ISW(1)))GO TO 1
            IF(IAND(IEEOF,MASK).EQ.IAND(MASK,ISW(1)))GO TO 2
            NERR=1
            WRITE(5,1000)ISW(1)
      1000  FORMAT(1X,'FILE SEARCH ERROR' O9/)
            RETURN
      2     CALL GETADR(MT1(1),NBUF(1))
            CALL WTQIO(IORLB,3,1,0,ISW,MT1,DSW)
            IF(IOSUC.EQ.IAND(MASK,ISW(1)))GO TO 1
            IF(IAND(IEEOF,MASK).EQ.IAND(MASK,ISW(1))) GO TO 3
            NERR=1
            RETURN
      3     MT1(1)=-1
            CALL WTQIO(IOSPF,3,1,0,ISW,MT1,DSW)
            RETURN
            END
```

PROGRAM SECTIONS

NAME	SIZE		ATTRIBUTES
$CODE1	000254	86	RW,I,CON,LCL
$PDATA	000014	6	RW,D,CON,LCL
$IDATA	000076	31	RW,D,CON,LCL
MTAPE3	012260	2648	RW,D,OVR,GBL
MTAPE5	000064	26	RW,D,OVR,GBL

TOTAL SPACE ALLOCATED = 012732 2797

NO FPP INSTRUCTIONS GENERATED

```
      C
                    SUBROUTINE EOFW(NERR)
                    IMPLICIT INTEGER(A-Z)
                    COMMON/MTAPE5/MASK,ISW(2),IOATT,IOSUC,IEALN,IORWD,
                1   IOWLB,IEVER,IOSPF,IEEOF,IOEOF,IORLB,MT0(6),MT1(6),DSW
                    NERR=0
                    DO 1 I=1,2
      1             CALL WTQIO(IOEOF,3,1,0,ISW(1))
                    IF(IOSUC.EQ.IAND(MASK,ISW(1)))GO TO 2
                    NERR=1
                    RETURN
      2             MT1(1)=-1
                    CALL WTQIO(IOSPF,3,1,0,ISW,MT1,DSW)
                    IF(IOSUC.EQ.IAND(MASK,ISW(1)))RETURN
                    NERR=1
                    RETURN
                    END
```

PROGRAM SECTIONS

NAME	SIZE		ATTRIBUTES
$CODE1	000152	53	RW,I,CON,LCL
$PDATA	000014	6	RW,D,CON,LCL
$IDATA	000034	14	RW,D,CON,LCL
$VARS	000002	1	RW,D,CON,LCL
MTAPE5	000064	26	RW,D,OVR,GBL

TOTAL SPACE ALLOCATED = 000310 100

NO FPP INSTRUCTIONS GENERATED

AD-A092 010 GTE PRODUCTS CORP NEEDHAM HEIGHTS MA COMMUNICATION S--ETC F/G 9/4
SPEECH ALGORITHM OPTIMIZATION AT 16 KBPS.(U)
SEP 80 R S CHEUNG, S Y KWON, A J GOLDBERG DCA100-79-C-0038

UNCLASSIFIED NL

3 OF 3
AD A
092010

END
DATE
FILMED
1-81
DTIC

```
C       SUBROUTINE INFO.FTN
C
        SUBROUTINE INFO
        IMPLICIT INTEGER(A-Z)
        COMMON/MTAPE0/NIN(256),NOUT(256)
        COMMON/MTAPE1/NSKIP,IST,NTOTI,NTUPS,NTOTO
        COMMON/MTAPE2/NEND,NERR,NFILE,NINS,NOUTS
        COMMON/MTAPE3/NBF(1324),NBUF(1324)
        LOGICAL*1 Y,ANS
        LOGICAL*1 EXTIN,EXTOUT,APPEND
        DIMENSION EXTIN(3),EXTOUT(3)
        DATA EXTIN/'I','N','P'/
        DATA EXTOUT/'O','U','T'/
        DATA Y/'Y'/
        DATA NEND,NERR,NFILE/0,0,0/
        DATA NSKIP,IST/1,1/
C
C       GET I/O INFORMATION  FOR SPEECH HANDLER
C
        APPEND=.FALSE.
        TYPE 1
1       FORMAT(1H$,'IS THE INPUT ON MAG. TAPE? ')
        READ(5,2)ANS
2       FORMAT(A1)
        IF(ANS.NE.Y)NINS=1
        TYPE 3
3       FORMAT(1H$'IS THE OUTPUT GOING TO MAG TAPE? ')
        READ (5,2) ANS
        IF(ANS.NE.Y) NOUTS=1
        IF(NOUTS.NE.0) GO TO 5
        TYPE 4
4       FORMAT(1H$,'APPEND DATA? ')
        READ (5,2) ANS
        IF(ANS.EQ.Y) APPEND=.TRUE.
5       IF(NOUTS.EQ.0) GO  TO 151
C       RSX11  SUPPORTED FILE
        TYPE 150
150     FORMAT(1H$,'OUTPUT FILE NAME= ')
        CALL FILEN(3,EXTOUT)
C
C       BEGINNING OF INPUT
C
151     IF(NINS.NE.0) GO TO 155
        TYPE 100
100     FORMAT(1H$'MT FILE NO.=(I3) ')
        READ(5,101)NFILE
101     FORMAT(I3)
        GO TO 14
155     TYPE 13
13      FORMAT(1H$,'INPUT FILE NAME= ')
        CALL FILEN(2,EXTIN)
        NFILE=1
14      CALL TAPE2(3)
        IF(NERR.NE.0) RETURN
        IF(APPEND)CALL TAPE2(4)
        RETURN
```

```
0048            END


PROGRAM SECTIONS

 NAME        SIZE                      ATTRIBUTES

$CODE1  000470   156                   RW,I,CON,LCL
$PDATA  000014     6                   RW,D,CON,LCL
$IDATA  000270    92                   RW,D,CON,LCL
$VARS   000012     5                   RW,D,CON,LCL
MTAPE0  002000   512                   RW,D,OVR,GBL
MTAPE1  000012     5                   RW,D,OVR,GBL
MTAPE2  000012     5                   RW,D,OVR,GBL
MTAPE3  012260  2648                   RW,D,OVR,GBL


TOTAL SPACE ALLOCATED = 015312  3429

NO FPP INSTRUCTIONS GENERATED
```

```
C
      SUBROUTINE FILEN(UNIT,EXT)
C
C     THIS SUBROUTINE ACCEPTS THE .'AME OF THE INPUT OR OUTPUT FILE
C     FORM THE TTY DEVICE 5
C     DEFAULT DEVICE
C     UNLESS SPECIFIED IN INPUT STRING
C     UNIT=UNIT NUMBER
C     EXT = LOGICAL*1 BUFFER OF EXTENSION
C
      IMPLICIT INTEGER(A-Z)
      LOGICAL*1 INSTR,DOT,BLNK,EXT
      DIMENSION INSTR(40)
      DIMENSION EXT(3)
      DATA BLNK,DOT/' ','.'/
C
C     INPUT FILE
152   READ (5,99)(INSTR(I),I=1,40)
99    FORMAT(40A1)
C     CHECK FOR END OF LINE
      DO 1600 I=40,1,-1
      J=I
1600  IF(INSTR(I).NE.BLNK)GO TO 1601
      TYPE 151
151   FORMAT(1H$,'>')
      GO TO 152
1601  DO 1602 I=1,J
      IF(INSTR(I).NE.BLNK) GO TO 1602
C
C
C     BLANK DISCOVERED-COLLAPSE LINE BY ONE AND DECREASE CHARACTER COU
C
      DO 1603 K=I,J-1
1603  INSTR(K)=INSTR(K+1)
      INSTR(J)=BLNK
      J=J-1
      GO TO 1601
1602  CONTINUE
      DO 103 I=1,J
103   IF(INSTR(I).EQ.DOT) GO TO 25
      INSTR(J+1)=DOT
      INSTR(J+2)=EXT(1)
      INSTR(J+3)=EXT(2)
      INSTR(J+4)=EXT(3)
      J=J+4
25    CALL SCAN(INSTR,J)
      CALL ASSIGN(UNIT,INSTR,J)
      RETURN
      END
```

PROGRAM SECTIONS

NAME	SIZE		ATTRIBUTES
$CODE1	000444	146	RW,I,CON,LCL

```
$IDATA   000042    17         RW,D,CON,LCL
$VARS    000060    24         RW,D,CON,LCL
$TEMPS   000004     2         RW,D,CON,LCL

TOTAL SPACE ALLOCATED = 000572   189

NO FPP INSTRUCTIONS GENERATED
```

```
C
      SUBROUTINE SCAN(BUF,LTH)
      IMPLICIT INTEGER(A-Z)
      LOGICAL*1 BUF,DEVICE
      DIMENSION BUF(1),DEVICE(4)
      DATA DEVICE/'S','Y','0',':'/
      DO 1 I=1,LTH
1     IF(BUF(I).EQ.DEVICE(4))RETURN
      LTH=LTH+4
      DO 2 I=LTH,5,-1
2     BUF(I)=BUF(I-4)
      DO 3 I=1,4
3     BUF(I)=DEVICE(I)
      RETURN
      END
```

PROGRAM SECTIONS

NAME	SIZE		ATTRIBUTES
$CODE1	000172	61	RW,I,CON,LCL
$IDATA	000012	5	RW,D,CON,LCL
$VARS	000006	3	RW,D,CON,LCL
$TEMPS	000002	1	RW,D,CON,LCL

TOTAL SPACE ALLOCATED = 000214 70

NO FPP INSTRUCTIONS GENERATED

TAPE2,LP:=TAPE2/NOTR

```
C       FFTRR8.FTN
C       JAN.,30, 1979
C       EQU. ROUTINE OF [100,117]FFTRR8
        SUBROUTINE FFTRR8(XR,XI,M,IS)
        DIMENSION XR(1),XI(1)
        N=2**M
        NV2=N/2
        NM1=N-1
        J=1
        DO 7 I=1,NM1
        IF(I.GE.J)GO TO 5
        T=XR(J)
        XR(J)=XR(I)
        XR(I)=T
        T=XI(J)
        XI(J)=XI(I)
        XI(I)=T
5       K=NV2
6       IF(K.GE.J)GO TO 7
        J=J-K
        K=K/2
        GO TO 6
7       J=J+K
        PI=3.1415927
        DO 20 L=1,M
        LE=2**L
        LE1=LE/2
        UR=1.0
        UI=0.0
        WR=COS(PI/LE1)
        WI=SIN(PI/LE1)
        IF(IS.LT.0)WI=-WI
        DO 20 J=1,LE1
        DO 10 I=J,N,LE
        IP=I+LE1
        TR=XR(IP)*UR-XI(IP)*UI
        TI=XR(IP)*UI+XI(IP)*UR
        XR(IP)=XR(I)-TR
        XI(IP)=XI(I)-TI
        XI(I)=XI(I)+TI
10      XR(I)=XR(I)+TR
        URR=UR*WR-UI*WI
        UI=UR*WI+WR*UI
20      UR=URR
        IF(IS.EQ.-1)GO TO 40
        DO 30 I=1,N
        XR(I)=XR(I)/FLOAT(N)
        XI(I)=XI(I)/FLOAT(N)
30      CONTINUE
40      RETURN
        END
```

PROGRAM SECTIONS

NAME SIZE ATTRIBUTES

```
$CODE1   001154   310            RW,I,CON,LCL
$PDATA   000004     2            RW,D,CON,LCL
$IDATA   000024    10            RW,D,CON,LCL
$VARS    000070    28            RW,D,CON,LCL
$TEMPS   000012     5            RW,D,CON,LCL

TOTAL SPACE ALLOCATED = 001306   355

FFTRR8,LP:=FFTRR8/NOTR
```

```
C       SER.FTN
C       SERIALIZE TRANSMITION PARAMETERS INTO BINARY DATA
C
C       AUG.4, 1980
C
        SUBROUTINE SER(INBA,INB,IBIL)
        COMMON/MTAPEO/NIN(170),NOUT(170)
        COMMON/SBTA/IBPT,IBBT,IBQL,IBQH,IBPL(4),IBPH(4)
        COMMON/SDTA/MPIT,IBETA,IQQL,IQQH,IDPL(4),IDPH(4)
        DIMENSION INBA(1),INB(1)
C
C       INI. THE TRANSMITER DATA
C
        DO 100 I=1,360
100     INBA(I)=0
C
C       SERIALIZE MPIT
        CALL DBCONV(MPIT,IBPT,INB)
        IQI=0
        DO 220 I=1,IBPT
        IQI=IQI+1
220     INBA(IQI)=INB(I)
C
C       SERIALIZE FOR BETA
        CALL DBCONV(IBETA,IBBT,INB)
        DO 230 I=1,IBBT
        IQI=IQI+1
230     INBA(IQI)=INB(I)
C
C       SERIALE FOR QQL
        CALL DBCONV(IQQL,IBQL,INB)
        DO 200 I=1,IBQL
        IQI=IQI+1
200     INBA(IQI)=INB(I)
C
C       SERIALIOZE FOR QQH
        CALL DBCONV(IQQH,IBQH,INB)
        DO 210 I=1,IBQH
        IQI=IQI+1
210     INBA(IQI)=INB(I)
C
C       SERIALIZE FOR LOW BAND PARCOR
C
        DO 240 J=1,4
        IBT=IBPL(J)
        CALL DBCONV(IDPL(J),IBT,INB)
        DO 250 I=1,IBT
        IQI=IQI+1
250     INBA(IQI)=INB(I)
240     CONTINUE
C
C       SERIALIZE FOR HIGH BAND PARCOR
C
        DO 260 J=1,4
        IBT=IBPH(J)
        CALL DBCONV(IDPH(J),IBT,INB)
```

```
0035                DO 270 I=1,IBT
0036                IQI=IQI+1
0037      270       INBA(IQI)=INB(I)
0038      260       CONTINUE
          C
          C         IQI WILL BE 50
0039                IQS=50
          C
          C         PROTECT 56 BITS OF IMPORTANT BAND
          C
0040                IBIH=3-IBIL
0041                IF(IBIL.LE.1)GO TO 300
          C
          C         IBIL=2 OR 3
0042                IES=0
0043                IEF=72
0044                IBL=IBIL
0045                IBS=IBIH
0046                GO TO 400
0047      300       CONTINUE
0048                IES=72
0049                IEF=0
0050                IBL=IBIH
0051                IBS=IBIL
0052      400       CONTINUE
0053                IF(IBL.EQ.3)GO TO 600
          C
          C         IBL=2
          C
          C         SERIALIZE HIGHER ENERGY ERROR SIGNALS
0054                DO 520 I=1,72
0055                IJ=I+IES
0056                CALL DBCONV(NIN(IJ),IBL,INB)
0057                DO 530 J=1,IBL
0058                IQS=IQS+1
0059      530       INBA(IQS)=INB(J)
0060      520       CONTINUE
          C
          C         SERIALIZE LOW ENERGY BAND ERROR SIGNALS
          C
0061                DO 540 I=1,72
0062                IJ=I+IEF
0063                IQS=IQS+1
0064      550       INBA(IQS)=NIN(IJ)
0065      540       CONTINUE
0066                RETURN
0067      600       CONTINUE
          C
          C         IBL=3
          C         IBS=0
0068                DO 610 I=1,31
0069                IJ=I+IES
0070                CALL DBCONV(NIN(IJ),IBL,INB)
0071                DO 620 J=1,IBL
0072                IQS=IQS+1
0073      620       INBA(IQS)=INB(J)
```

```
0074     610     CONTINUE
         C
0075             DO 630 I=32,72
0076             IJ=I+IES
0077             CALL DBCONV(NIN(IJ),IBL,INB)
0078             DO 640 J=1,2
0079             IQS=IQS+1
0080     640     INBA(IQS)=INB(J)
         C       STORE THE THIRD BITS OUT OF PROTECTION GROUP
0081             INBA(194+I)=INB(3)
0082     630     CONTINUE
0083             RETURN
0084             END
```

PROGRAM SECTIONS

NAME	SIZE		ATTRIBUTES
$CODE1	001716	487	RW,I,CON,LCL
$IDATA	000104	34	RW,D,CON,LCL
$VARS	000026	11	RW,D,CON,LCL
$TEMPS	000004	2	RW,D,CON,LCL
MTAPEO	001250	340	RW,D,OVR,GBL
SBTA	000030	12	RW,D,OVR,GBL
SDTA	000030	12	RW,D,OVR,GBL

TOTAL SPACE ALLOCATED = 003404 898

NO FPP INSTRUCTIONS GENERATED

SER,LP:=SER/NOTR

```
          C        ENCBCH.FTN
          C        MARCH 16, 1979
          C        ENCODING OF A(63,45)BCH CODE
0001               SUBROUTINE ENCBCH(INBA,INA,INC,KT)
0002               DIMENSION INBA(1),INA(1),INC(1),ING(19)
0003               DATA ING/1,1,1,1,0,0,0,0,0,1,0,1,1,0,0,1,1,1,1/
          C        CALCULATE PARITY BITS
0004               DO 10 I=1,63
0005               KTI=(KT-1)*63+I
0006               INA(I)=INBA(KTI)
0007               IF(I.GT.45)INA(I)=0
0008      10       CONTINUE
0009               CALL GF2DIV(INA,63,ING,19,INC,NC)
          C        SHIFT 18 BITS FOR PARITY BITS
          C
0010               KTF=270+18*KT
0011               KTI=KTF-63*KT
0012               DO 30 I=1,KTI
0013               INBA(KTF+1-I)=INBA(KTF-17-I)
0014      30       CONTINUE
          C
          C        STORE PARITY BITS
0015               DO 20 I=1,NC
0016               KTI=(KT-1)*63+I+45
0017      20       INBA(KTI)=INC(I)
0018               RETURN
0019               END
```

PROGRAM SECTIONS

NAME	SIZE		ATTRIBUTES
$CODE1	000412	133	RW,I,CON,LCL
$PDATA	000010	4	RW,D,CON,LCL
$IDATA	000054	22	RW,D,CON,LCL
$VARS	000056	23	RW,D,CON,LCL
$TEMPS	000002	1	RW,D,CON,LCL

TOTAL SPACE ALLOCATED = 000556 183

NO FPP INSTRUCTIONS GENERATED

```
C       CEIR.FTN
C       MARCH 16, 1979
C       CHANNEL ERROR SIMULATION ROUTINE
        SUBROUTINE CEIR(INBA,NBRPF,PROB,IRN,JRN,NERB,NEPB)
        COMMON/SW/ICOUNT,IPRSW
        DIMENSION INBA(1),NERB(1)
        NEPB1=NEPB+1
        DO 50 I=1,NEPB1
50      NERB(I)=0
C       XMIT INPUT BINARY VECTOR
        IF(NEPB.LE.0)GO TO 40
        DO 30 J=1,NEPB
        DO 10 I=1,63
        ISV1=NERB(J)
        IP=I+63*(J-1)
        CALL RANERR(INBA(IP),PROB,IRN,JRN,NERB(J))
        IF(ISV1.NE.NERB(J).AND.IPRSW.EQ.1)WRITE(4,100)ICOUNT,IP
10      CONTINUE
30      CONTINUE
40      CONTINUE
        ISTP=NEPB*63+1
        DO 20 I=ISTP,NBRPF
        ISV1=NERB(NEPB1)
        CALL RANERR(INBA(I),PROB,IRN,JRN,NERB(NEPB1))
        IF(ISV1.NE.NERB(NEPB1).AND.IPRSW.EQ.1)WRITE(4,100)ICOUNT,I
20      CONTINUE
100     FORMAT(1X,'FR=',I4,2X,'ERR LC=',I3)
        RETURN
        END
```

PROGRAM SECTIONS

NAME	SIZE		ATTRIBUTES
$CODE1	000664	218	RW,I,CON,LCL
$IDATA	000066	27	RW,D,CON,LCL
$VARS	000014	6	RW,D,CON,LCL
$TEMPS	000010	4	RW,D,CON,LCL
SW	000004	2	RW,D,OVR,GBL

TOTAL SPACE ALLOCATED = 001002 257

NO FPP INSTRUCTIONS GENERATED

```
C       RANERR.FTN
        SUBROUTINE RANERR(IX,PROB,IRN,JRN,NER)
        CALL RANDU(IRN,JRN,YQR)
        IF(YQR.GE.PROB)RETURN
        IX=IEOR(IX,1)
        NER=NER+1
        RETURN
        END
```

PROGRAM SECTIONS

NAME	SIZE		ATTRIBUTES
$CODE1	000066	27	RW,I,CON,LCL
$IDATA	000010	4	RW,D,CON,LCL
$VARS	000004	2	RW,D,CON,LCL

TOTAL SPACE ALLOCATED = 000102   33

```
C       DECBCH.FTN
C       DECODING OF BCH CODE
C       EXAMPLE OF A(63,45) BCH CODE WHICH CORRECT 3 ERRORS
C       G(X)=(X**6+X+1)(X**6+X**4+X**2+X+1)(X**6+X**5+X**2+X+1)
        SUBROUTINE DECBCH(INBA,INA,KT,NES)
        COMMON/SW/ICOUNT,IPRSW
        DIMENSION ISD1(7),ISD3(7),ISD5(7),INA(1),INB(63),INC(7),IM1(7)
        DIMENSION INBA(1),NERL(3)
        DATA IM1/1,0,0,0,0,1,1/
C       READ INPUT VECTOR INA
        DO 10 I=1,63
        KTI=(KT-1)*45+I
10      INA(I)=INBA(KTI)
C       START DECODING
C
C       DECODING ROUTINE
C
C       ***********************************
C       CALCULATE POWER SUMS
        CALL GF2DIV(INA,63,IM1,7,ISD1,NC)
C       CALCULATE R(X**3)
        DO 18 I=1,63
18      INB(I)=0
        DO 19 I=1,63
        I3=(63-I)*3
        IR=I3-(I3/63)*63
        IT=63-IR
        IX=INA(I)
19      INB(IT)=IEOR(IX,INB(IT))
        CALL GF2DIV(INB,63,IM1,7,ISD3,NC)
C       CALCULATE R(X**5)
        DO 28 I=1,63
28      INB(I)=0
        DO 29 I=1,63
        I5=(63-I)*5
        IR=I5-(I5/63)*63
        IT=63-IR
        IX=INA(I)
29      INB(IT)=IEOR(IX,INB(IT))
        CALL GF2DIV(INB,63,IM1,7,ISD5,NC)
101     FORMAT(1X,'S1=',18I1)
202     FORMAT(1X,'S3=',18I1)
303     FORMAT(1X,'S5=',18I1)
C       CHECK ERROR RANGE
        DO 40 I=1,6
        IF(ISD1(I).EQ.1)GO TO 50
        IF(ISD3(I).EQ.1)GO TO 50
        IF(ISD5(I).EQ.1)GO TO 50
40      CONTINUE
C       NO CHANNEL ERROR
        GO TO 998
50      CONTINUE
C       CORRECT CHANNEL ERROR
C
C       CALCULATE SIGMA(I),I=1,3
C       CALCULATE DET(3),I.E.,S1**3+S3
```

```
0038                CALL GF2MUL(ISD1,6,ISD1,6,INC,ND,IM1,7)
0039                CALL GF2MUL(ISD1,6,INC,6,INA,ND,IM1,7)
0040                CALL GF2ADD(INA,6,ISD3,6,INB,ND)
          CC        IF(ISW0.EQ.2)WRITE(5,303)(INB(I),I=1,ND)
0041                NES=3
0042                DO 60 I=1,6
0043                IF(INB(I).EQ.1)GO TO 70
0044      60        CONTINUE
0045                NES=1
          C         ONLY ONE ERROR OCCUR
0046                GO TO 80
0047      70        CONTINUE
          C         CALCULATE SIGMA(2) AND SIGMA(3)
          C         STORE INB(I),I=1,6
0048                DO 90 I=1,6
0049      90        INA(I)=INB(I)
          C         CALCULATE SIGMA(2)
          C         CALCULATE S1**2*S3+S5
0050                CALL GF2MUL(INC,6,ISD3,6,INB,NC,IM1,7)
0051                CALL GF2ADD(INB,6,ISD5,6,INC,NC)
          CC        IF(ISW0.EQ.2)WRITE(5,303)(INC(I),I=1,6)
0052                DO 789 I=1,6
0053                IF(INC(I).EQ.1)GO TO 444
0054      789       CONTINUE
          C         INC=0
0055                GO TO 85
0056      444       CONTINUE
          C         FIND THE ORDER OF INC=S1**2*S3+S5
          C         FIND THE ORDER OF INA=S1**3+S3
          C         INI INB
0057                DO 39 I=1,63
0058      39        INB(I)=0
0059                IORA=0
0060                IORC=0
0061                DO 49 I=2,63
0062                INB(64-I)=1
0063                INB(65-I)=0
0064                CALL GF2DIV(INB,63,IM1,7,ISD3,NC)
          CC        IF(ISW0.EQ.2.AND.ISW1.EQ.1)WRITE(5,404)I,(ISD3(J),J=1,6)
0065      404       FORMAT(1X,'N=',I3,3X,8I1)
          C         CHECK THE ORDER
0066                DO 59 J=1,6
0067                IF(ISD3(J).NE.INA(J))GO TO 69
0068      59        CONTINUE
0069                IORA=I-1
0070      69        DO 79 J=1,6
0071                IF(ISD3(J).NE.INC(J))GO TO 89
0072      79        CONTINUE
0073                IORC=I-1
0074      89        IF(IORA.NE.0.AND.IORC.NE.0)GO TO 109
0075      49        CONTINUE
0076      109       IOR3=IORC-IORA
0077                IF(IOR3.LT.0)IOR3=IOR3+63
0078                IOR3=63-IOR3
0079                DO 119 I=1,63
0080                IT=0
```

```
0081                IF(I.EQ.IOR3)IT=1
0082      119       INB(I)=IT
          C         CALCULATE SIGMA(2)
0083                CALL GF2DIV(INB,63,IM1,7,ISD3,NC)
          C
          CC        IF(ISW0.EQ.2)WRITE(5,202)(ISD3(I),I=1,6)
          C         CALCULATE SIGMA(3)
0084                CALL GF2MUL(ISD1,6,ISD3,6,INC,NC,IM1,7)
0085                CALL GF2ADD(INA,6,INC,6,ISD5,NC)
          CC        IF(ISW0.EQ.2)WRITE(5,303)(ISD5(I),I=1,6)
          C         IF ISD5=0, THEN NES=2
0086                DO 71 I=1,6
0087                IF(ISD5(I).NE.0)GO TO 80
0088      71        CONTINUE
0089                NES=2
0090                GO TO 80
0091      85        CONTINUE
0092                DO 87 I=1,6
0093                ISD3(I)=0
0094                ISD5(I)=INA(I)
0095      87        CONTINUE
0096      80        CONTINUE
          C         CORRECT NES ERROR BY CHIEN'S SEARCH METHOD
          C
0097                NEST=0
0098                DO 11 II=1,63
0099                II1=II-1
0100                IF(II1.EQ.0)II1=63
          C         INI VECTOR C
0101                DO 22 I=1,6
0102      22        INB(I)=0
0103                CALL GF2ADD(ISD1,6,INB,6,INC,NC)
0104                IF(NES.EQ.1)GO TO 33
0105                CALL GF2ADD(ISD3,6,INC,6,INB,NB)
0106                CALL GF2ADD(INB,6,ISD5,6,INC,NC)
0107      33        CONTINUE
          CC        IF(ISW0.EQ.2.AND.ISW1.EQ.1)WRITE(5,303)(INC(I),I=1,6)
0108                IF(INC(6).EQ.0)GO TO 44
0109                DO 55 I=1,5
0110                IF(INC(I).EQ.1)GO TO 44
0111      55        CONTINUE
          C         CORRECT ERROR
0112                KTT=(KT-1)*63+II1
0113                NEST=NEST+1
0114                NERL(NEST)=KTT
0115      44        CONTINUE
          C         SHIFT ISV1,ISV3,ISV5
0116                INB(1)=1
0117                INB(2)=0
0118                DO 88 I=1,6
0119      88        INA(I)=ISD1(I)
0120                CALL GF2MUL(INA,6,INB,2,ISD1,NC,IM1,7)
0121                IF(NES.EQ.1)GO TO 11
0122                INB(1)=1
0123                INB(2)=0
0124                INB(3)=0
```

```
              DO 775 I=1,6
      775     INA(I)=ISD3(I)
              CALL GF2MUL(INA,6,INB,3,ISD3,NC,IM1,7)
              INB(1)=1
              INB(2)=0
              INB(3)=0
              INB(4)=0
              DO 665 I=1,6
      665     INA(I)=ISD5(I)
              CALL GF2MUL(INA,6,INB,4,ISD5,NC,IM1,7)
      11      CONTINUE
      C       CHECK ERROR STATUS
              IF(NES.EQ.NEST)GO TO 888
              NES=4
              GO TO 998
      C       CORRECT ERRORS
      888     CONTINUE
              DO 72 I=1,NES
              KTT=NERL(I)
              IF(IPRSW.EQ.1)WRITE(4,707)ICOUNT,KTT
      707     FORMAT(1X,'FR=',I6,2X,'EC LC=',I3)
              KTR=KTT-(KT-1)*18
              INBA(KTR)=IEOR(INBA(KTR),1)
      72      CONTINUE
      998     CONTINUE
      C
      C       COMPRESS 18 PARITY BITS FOR DESERIALIZATION ROUTINE
      C
              KTI=45*KT+1
              KTF=360-18*KT
              DO 900 I=KTI,KTF
      900     INBA(I)=INBA(I+18)
              RETURN
              END
```

PROGRAM SECTIONS

NAME	SIZE		ATTRIBUTES
$CODE1	002706	739	RW,I,CON,LCL
$PDATA	000030	12	RW,D,CON,LCL
$IDATA	000430	140	RW,D,CON,LCL
$VARS	000362	121	RW,D,CON,LCL
$TEMPS	000002	1	RW,D,CON,LCL
SW	000004	2	RW,D,OVR,GBL

TOTAL SPACE ALLOCATED = 003756 1015

NO FPP INSTRUCTIONS GENERATED

CESR,LP:=CESR/NOTR

```
          C       MRNSA.FTN
          C
          C       JUNE 6, 1980
          C
          C
          C
          C       ***************
          C
          C       NOISE SUPPRESION ROUTINE BY R.J. MCAULAY
          C
          C
          C R.J. MCAULAY,"SPEECH ENHANCEMENT USING A SOFT-DECISION NOISE
          C SUPPRESION FILTER,"IEEE TRANS. ASSP APRIL 1980.
          C
          C       ****************
          C
0001              SUBROUTINE MRNSA(XR,XI,NTOTI,NSF)
0002              COMMON/MTAPEO/NIN(170),NOUT(170)
0003              COMMON/NSTBL/FNSTBL(50)
0004              DIMENSION XR(1),XI(1)
0005              DIMENSION STCS(129),DIST(128)
0006              DATA STCS/129*1.0/
0007              DATA DIST/10*0.0,117*200.,0.0/
0008              DATA AGNO,AGNL/2*1.0/
          C       TAKE DFT OF NOISY INPUT SPEECH SIGNAL
          C
          C       CALCULATE ENERGY FOR V/N/S DECISION
          C
0009              ENO=0.0
0010              DO 8010 I=1,256
0011              XI(I)=0.0
0012              XR(I)=0.0
0013              IF(I.GT.NTOTI)GO TO 8010
0014              XR(I)=FLOAT(NIN(I))
0015      8010    ENO=ENO+XR(I)**2
0016              ENO=ENO/128.0
          C
          C
          C       PERFORM DFT
          C
0017              CALL FFTRR8(XR,XI,8,-1)
          C
          C
          C       CLASSIFY SIGNAL STATE
          C       USE ROBERT'S MODIFIED NOISE DETECTION ALGORITHM
          C
0018              CALL MRNDA(ENO,DIST,NTHO,THL,ENN,FMU)
          C
          C
          C
          C
          C       SUPPRESS NOISE IN FREQUENCY DOMAIN
          C
          C       USE MCAULAY'S APPROACH
          C
          C       START SUPPRESION
```

```
C
              AGN=0.0
              DO 8030 J=1,256
              JX=J
              IF(J.GE.129)JX=258-J
C             SIGNAL POWER SPECTRUM ASPW
              ASPW=XR(J)**2+XI(J)**2
              GN=ASPW-STCS(JX)
              IF(GN.GE.1.E-6)GN=GN/ASPW
              IF(GN.LE.1.E-6)GN=1.E-6
C
C             UPDATE NOISE SATTISTICS IF SPEECH NOT PRESENT
C
              IF(J.GT.129)GO TO 8033
              IF(NTHO.EQ.2)GO TO 8033
C
C             UPDATE NOISE STATISTICS
C
              TT=STCS(J)-ASPW
              IF(TT.LT.0.0)TT=TT*0.7788
              IF(TT.GE.0.0)TT=TT*0.875
              STCS(J)=ASPW+TT
      8033    CONTINUE
C             CALCULATE SUPPRESION FACTOR
C
C
              IGN=INT(GN*50.0)+1
              IF(IGN.GE.50)IGN=50
              GN=FNSTBL(IGN)
              AGN=AGN+GN
C             INSERT SMOOTHING GAIN PROGRAM IN LATER VERSION
C
      8037    XR(J)=XR(J)*GN
              XI(J)=XI(J)*GN
      8030    CONTINUE
C
C             NOISE SUPPRESSION IS DONE IN FREQUENCY DOMAIN
C
C             PERFORM INVERSE DFT
              CALL FFTRR8(XR,XI,8,1)
C
C             STORE NOISE SUPPRESSED OUTPUT TIME DOMAIN FOR SBAPC
C
C
              AGN=AGN/256.0
C
C             CALCULATE LONG TERM AGN AND USE IT TO ADJUST THE NOISE SUPPRESION
C             FACTOR ADAPTIVELY IN LATER VERSION
              AGNL=AGNL*127./128.0+AGN/128.0
C
C             CALCULATE SHORT TERM AGN
C
              AGNO=AGNO*0.75+AGN*0.25
              FMUL=AGNO/AGNL
              FMUL=SQRT(FMUL)
C
```

```
      C
0047            DO 8040 I=1,NTOTI
0048            XR(I)=FMUL*XR(I)
0049  8040      CONTINUE
      C
0050            RETURN
0051            END
```

PROGRAM SECTIONS

NAME	SIZE		ATTRIBUTES
$CODE1	001070	284	RW,I,CON,LCL
$PDATA	000024	10	RW,D,CON,LCL
$IDATA	000066	27	RW,D,CON,LCL
$VARS	002072	541	RW,D,CON,LCL
$TEMPS	000002	1	RW,D,CON,LCL
MTAPE0	001250	340	RW,D,OVR,GBL
NSTBL	000310	100	RW,D,OVR,GBL

TOTAL SPACE ALLOCATED = 005056 1303

```
      C       MRNDA.FTN
      C
      C       MAY 15, 1980
      C
      C       MODIFIED ROBERT'S NOISE DETCTION ALGORITHM(ASSP 144, APRIL 1980)
      C
      C
0001          SUBROUTINE MRNDA(ENO,DIST,NTH,THL,FJMX,FMU)
0002          DIMENSION DIST(1)
0003          DATA THMXT,THMX,THOL,THOMN/2*32768.0,16384.0,1024./
0004          DATA FMUL/1.0/
      C
      C       CHECK INPUT ENERGY
      C
      C       NTH=0:SILENCE
      C       NTH=1:NOISE PRESENT
      C       NTH=2:SPEECH PRESENT
      C
      C       SLOW DEGARDATION OF MAX ENERGY VALUE
      C
0005          THMX=THMX*0.995
      C       CALCULATE AVERAGE FRAME ENERGY
0006          THMXT=THMXT*0.75+ENO*0.25
0007          IF(THMXT.GE.THMX)THMX=THMXT
      C       FIND MAX DIST. POINT
0008          JMX=1
0009          DIMX=DIST(1)
0010          DO 100 I=2,128
0011          IF(DIST(I).LE.DIMX)GO TO 100
0012          JMX=I
0013          DIMX=DIST(I)
0014  100     CONTINUE
      C
      C       FIND AVERAGE VALUE OF THMX AND JMX EQUIVALENT
      C
0015          FJMX=256.0*FLOAT(JMX+1)/FMUL
      C
      C       DECREASE FMUL IF MORE THAN 100 FRAMES ARE SPEECH'S
0016          IF(NTH.NE.2)ICT=0
0017          IF(NTH.EQ.2)ICT=ICT+1
0018          IF(ICT.LE.50)GO TO 333
0019          FMUL=FMUL/2.0
0020          GO TO 222
0021  333     IF(JMX.NE.1)GO TO 222
      C
      C       INCREASE FMUL
0022          FMUL=FMUL*2.0
0023  222     CONTINUE
0024          AVENN=SQRT(FJMX*THMX)
0025          FMULT=32768.0/AVENN
      C       SLOW CHANGE OF FMUL TIME CONST 1 SEC
      C
0026          FMUL=FMULT+0.875*(FMUL-FMULT)
0027          FMU=FMUL
0028          NTH=2
      C       TEST FOR NOISE PRESENSE
```

```
              C
              C        DECAY HISTOGRAM BY 4-SEC TIME CONSTATNT
              C        FS=6400, FRAME SIZE=22.5 MS(144 SAMPLES)
              C
0029                   FNORM=0.0
0030                   DO 10 I=1,128
0031                   DIST(I)=DIST(I)*0.9944
0032                   FNORM=FNORM+DIST(I)
0033          10       CONTINUE
              C
              C        NORMAIZE ENERGY SUCH THAT THOL=16384.
              CC       FMUL=1.0
0034                   EN=ENO*FMUL
0035                   IF(EN.GE.32768.)RETURN
              C        CALCULATE PRESENT FRAME ENERGY BIN
              C
0036                   JBIN=INT(EN/256.)
0037                   NTH=0
0038                   DECAY=0.60653
0039                   THNW=0.0
0040                   IF(JBIN.EQ.0)GO TO 30
0041                   DIST(JBIN)=DIST(JBIN)+144.
0042                   FNORM=FNORM+144.
              C
              C        SEARCH FOR SILENCE AND SET NEW THRESHOLD
              C
0043                   THSL=FNORM/4.0
0044                   THNC=FNORM*0.8
0045                   THMV=0.0
0046                   DO 20 J=1,10
0047          20       THMV=THMV+DIST(J)
0048                   IF(THMV.GE.THSL)GO TO 30
              C
              C        NOISE PRESENT
              C
0049                   NTH=1
              C
0050                   DO 40 J=11,128
0051                   THMV=THMV+DIST(J)
0052                   IF(THMV.GE.THNC)GO TO 50
0053          40       CONTINUE
0054          50       CONTINUE
0055                   THNW=256.*FLOAT(J)
0056                   IF(EN.GE.THOL)NTH=2
0057                   IF(THNW.LT.THOL)GO TO 30
0058                   DECAY=0.945
0059          30       CONTINUE
0060                   THOL=THNW+DECAY*(THOL-THNW)
              CC       THOL=THOL+DECAY*(THNW-THOL)
0061                   IF(THOL.LE.THOMN)THOL=THOMN
0062                   IF(NTH.EQ.0.AND.THOL.LE.EN)THOL=EN
0063                   THL=THOL/FMUL
0064                   RETURN
0065                   END
```

```
PROGRAM SECTIONS

 NAME         SIZE                       ATTRIBUTES

$CODE1   001232    333                   RW,I,CON,LCL
$PDATA   000024     10                   RW,D,CON,LCL
$IDATA   000012      5                   RW,D,CON,LCL
$VARS    000106     35                   RW,D,CON,LCL


TOTAL SPACE ALLOCATED = 001376    383

BNSR,LP:=BNSR/NOTR
```

```
C       DSER.FTN
C       DESERIALIZE BINARY DATA
C       AUG 4 ,1980
        SUBROUTINE DSER(INBA,INB,QQL,QQH,IBIL,IBIH,ILMX,ILMN)
        COMMON/MTAPEO/NIN(170),NOUT(170)
        COMMON/SBTA/IBPT,IBBT,IBQL,IBQH,IBPL(4),IBPH(4)
        COMMON/SDTA/MPIT,IBETA,IQQL,IQQH,IDPL(4),IDPH(4)
        DIMENSION INBA(1),INB(1)
C
C       DESERIALIZE PITCH
C
        IQI=0
        DO 220 I=1,IBPT
        IQI=IQI+1
220     INB(I)=INBA(IQI)
        CALL BDCONV(INB,IBPT,MPIT)
C
C       DESERIALE FOR BETA
        DO 230 I=1,IBBT
        IQI=IQI+1
230     INB(I)=INBA(IQI)
        CALL BDCONV(INB,IBBT,IBETA)
C
C       DESERIALIZE FOR QQL
        DO 200 I=1,IBQL
        IQI=IQI+1
200     INB(I)=INBA(IQI)
        CALL BDCONV(INB,IBQL,IQQL)
C
C       DEQUANTIZE FOR QQL
        IQD=32
        CALL DEQTZ(IQD,IQQL,QQL)
        IQD=IQD+2**(IBQL+1)-1
C
C       DESERIALIZE FOR IQQH
        DO 210 I=1,IBQH
        IQI=IQI+1
210     INB(I)=INBA(IQI)
        CALL BDCONV(INB,IBQH,IQQH)
C
C       DEQUNTIZE FOR QQH
C
        CALL DEQTZ(IQD,IQQH,QQH)
C
C       FIND THE BITS ASSIGNMENTS FOR LOW BAND AND HIGH BAND
C
C       ASSUME THE AVERAGE BITS=1.5
C
C       QQL IS LOG QQL OF BASE 2
        FIBIL=1.5+(QQL-QQH)/2.0
        IBIL=FIBIL+0.5
        IF(IBIL.GE.ILMX)IBIL=ILMX
        IF(IBIL.LE.ILMN)IBIL=ILMN
        IBIH=3-IBIL
        QQL=2.0**QQL
        QQH=2.0**QQH
```

```
C
C        DESERIALIZE FOR PARCOR OF LOW BAND
C
         DO 240 J=1,4
         IBT=IBPL(J)
         DO 250 I=1,IBT
         IQI=IQI+1
250      INB(I)=INBA(IQI)
         CALL BDCONV(INB,IBT,IDPL(J))
240      CONTINUE
C
C        DESERIALIZE FOR PARCOR OF HIGH BAND
C
         DO 260 J=1,4
         IBT=IBPH(J)
         DO 270 I=1,IBT
         IQI=IQI+1
270      INB(I)=INBA(IQI)
         CALL BDCONV(INB,IBT,IDPH(J))
260      CONTINUE
C
C        DESERIALIZE FORERROR SIGNALS
         IF(IBIL.LE.1)GO TO 300
C        IBIL=2 OR 3
C
         IES=0
         IEF=72
         IBL=IBIL
         IBS=IBIH
         GO TO 400
300      CONTINUE
         IES=72
         IEF=0
         IBL=IBIH
         IBS=IBIL
400      CONTINUE
C
C
         IQS=50
C
         IF(IBL.EQ.3)GO TO 600
C
C        IBL=2
C        DESERIALIZE FOR HIGHER ENERGY BAND
C
C
         DO 520 I=1,72
         IJ=I+IES
         DO 530 J=1,IBL
         IQS=IQS+1
530      INB(J)=INBA(IQS)
         CALL BDCONV(INB,IBL,NIN(IJ))
520      CONTINUE
C
C        DESERIALIZE LESS IMPORTANT BAND
C
```

```
0069                 DO 540 I=1,72
0070                 IJ=I+IEF
0071                 IQS=IQS+1
0072                 NIN(IJ)=INBA(IQS)
0073      540        CONTINUE
0074                 RETURN
0075      600        CONTINUE
          C
          C          IBL=3
          C
0076                 DO 610 I=1,31
0077                 IJ=I+IES
0078                 DO 620 J=1,IBL
0079                 IQS=IQS+1
0080      620        INB(J)=INBA(IQS)
0081                 CALL BDCONV(INB,IBL,NIN(IJ))
0082      610        CONTINUE
          C
0083                 DO 630 I=32,72
0084                 IJ=I+IES
0085                 DO 640 J=1,2
0086                 IQS=IQS+1
0087      640        INB(J)=INBA(IQS)
0088                 INB(3)=INBA(194+I)
0089                 CALL BDCONV(INB,IBL,NIN(IJ))
0090      630        CONTINUE
          C
          C          END OF DESERIALIZATION
0091                 RETURN
0092                 END
```

PROGRAM SECTIONS

NAME	SIZE		ATTRIBUTES
$CODE1	002162	569	RW,I,CON,LCL
$IDATA	000124	42	RW,D,CON,LCL
$VARS	000032	13	RW,D,CON,LCL
$TEMPS	000004	2	RW,D,CON,LCL
MTAPE0	001250	340	RW,D,OVR,GBL
SBTA	000030	12	RW,D,OVR,GBL
SDTA	000030	12	RW,D,OVR,GBL

TOTAL SPACE ALLOCATED = 003674   990

DSER,LP:=DSER/NOTR

```
C       GF2ADD.FTN
C       ADDITION OVER GF(2)
C       POLINOMIAL A(X) MUST BE ORDERED IN DESCENDING POWER SERIES
        SUBROUTINE GF2ADD(INA,NA,INB,NB,INC,NC)
        DIMENSION INA(1),INB(1),INC(1)
        NC=NA
        IF(NB.GT.NA)NC=NB
        DO 10 I=1,NC
        IC=NC+1-I
        IRA=NA+1-I
        IRB=NB+1-I
        ITA=0
        ITB=0
        IF(IRA.GT.0)ITA=INA(IRA)
        IF(IRB.GT.0)ITB=INB(IRB)
        INC(IC)=IEOR(ITA,ITB)
 10     CONTINUE
        RETURN
        END
```

PROGRAM SECTIONS

NAME	SIZE		ATTRIBUTES
$CODE1	000272	93	RW,I,CON,LCL
$IDATA	000036	15	RW,D,CON,LCL
$VARS	000014	6	RW,D,CON,LCL
$TEMPS	000002	1	RW,D,CON,LCL

TOTAL SPACE ALLOCATED = 000346 115

NO FPP INSTRUCTIONS GENERATED

```
FORTRAN IV-PLUS V02-51E          16:45:06     06-OCT-80              PAGE 2
GF2AMD.FTN          /WR

        C       GF2MUL.FTN
        C       MULTIPLICATION OVER GF(2)
        C       NA<NF,NB<NF
0001            SUBROUTINE GF2MUL(INA,NA,INB,NB,INC,NC,INF,NF)
0002            DIMENSION IAT(17),INA(1),INB(1),INC(1),INF(1)
0003            NCC=NA+NB-1
        C       INI VECTOR C
0004            DO 10 I=1,NCC
0005    10      IAT(I)=0
        C       MULTIPLY A AND B
0006            DO 20 I=1,NA
0007            DO 30 J=1,NB
0008            IC=I+J-1
0009            IT=IAND(INA(I),INB(J))
0010            IAT(IC)=IEOR(IAT(IC),IT)
0011    30      CONTINUE
0012    20      CONTINUE
0013            CALL GF2DIV(IAT,NCC,INF,NF,INC,NC)
0014            RETURN
0015            END
```

```
PROGRAM SECTIONS

 NAME       SIZE                 ATTRIBUTES

$CODE1  000344   114             RW,I,CON,LCL
$IDATA  000066    27             RW,D,CON,LCL
$VARS   000054    22             RW,D,CON,LCL
$TEMPS  000004     2             RW,D,CON,LCL

TOTAL SPACE ALLOCATED = 000512   165

NO FPP INSTRUCTIONS GENERATED
```

```
C     GF2DIV.FTN
C     FINITE FIELD DIVISION
C     DEVISOR VECTOR B IS DESTROYED IN COMPUTATION IF NOT NORMALIZWD
      SUBROUTINE GF2DIV(INA,NA,INB,NB,INC,NC)
      DIMENSION INA(1),INB(1),INC(1)
C     NORMALIZE VECTOR B
      NC=NB-1
      NBP=NB
      DO 11 I=1,NB
      IF(INB(1).EQ.1)GO TO 22
      NBP=NBP-1
      IF(NBP.LE.0)GO TO 11
      DO 33 J=1,NBP
33    INB(J)=INB(J+1)
11    CONTINUE
C     VECTOR B=0
      WRITE(5,100)
100   FORMAT(1X,'DIVISOR=0')
      RETURN
22    CONTINUE
      IF(NA.GE.NBP)GO TO 10
C     INA(I) IS THE ANSWER
      DO 20 I=1,NC
      IR=NC+1-I
      INC(IR)=0
      IF(I.GT.NA)GO TO 20
      ITA=NA+1-I
      INC(IR)=INA(ITA)
20    CONTINUE
      RETURN
10    CONTINUE
C     INI VECTOR C
C     ACTUAL NA MAY BE SMALLER THAN NBP
      DO 30 I=1,NBP
30    INC(I)=INA(I)
      NAP=NBP
111   CONTINUE
C     CHECK C(1)=1
      IF(INC(1).EQ.0)GO TO 222
C     START DIVISION
      DO 50 I=1,NBP
50    INC(I)=IEOR(INC(I),INB(I))
222   CONTINUE
      NAP=NAP+1
      IF(NAP.GT.NA)GO TO 333
C     SHIFT ONE BIT LEFT
      DO 60 I=1,NBP-1
60    INC(I)=INC(I+1)
      INC(NBP)=INA(NAP)
      GO TO 111
333   CONTINUE
C     INSERT 0 IF NBP NOT EQUAL TO NB
      IF(NBP.NE.NB)GO TO 777
      DO 555 I=1,NC
555   INC(I)=INC(I+1)
      RETURN
```

```
777     CONTINUE
        DO 773 I=1,NC
        IT=0
        IR=NC+1-I
        IBC=NBP+1-I
        IF(I.GT.NBP)GO TO 773
        IT=INC(IBC)
773     INC(IR)=IT
        RETURN
        END
```

PROGRAM SECTIONS

NAME	SIZE		ATTRIBUTES
$CODE1	001004	258	RW,I,CON,LCL
$IDATA	000054	22	RW,D,CON,LCL
$VARS	000020	8	RW,D,CON,LCL
$TEMPS	000010	4	RW,D,CON,LCL

TOTAL SPACE ALLOCATED = 001110 292

NO FPP INSTRUCTIONS GENERATED

GF2AMD,LP:=GF2AMD/NOTR

```
FORTRAN IV-PLUS V02-51E          16:45:40    06-OCT-80          PAGE 1
CONV.FTN            /WR

        C       DBCONV.FTN
        C       MARCH 13, 1979
        C       DECIMAL TO BINARY CONVERSION ROUTINE
        C       SUBROUTINE DBCONV(IX,LIB,INB)
        C       IX;INPUT INTERGER
        C       LIB;LENGTH OF OUTPUT VECTOR
        C       IX=INB(LIB)+INB(LIB-1)*2+ . . .
                SUBROUTINE DBCONV(IX,LIB,INB)
                DIMENSION INB(1)
                IY=IX
                DO 10 I=1,LIB
                IR=LIB+1-I
                INB(IR)=MOD(IY,2)
        10      IY=IY/2
                RETURN
                END

PROGRAM SECTIONS

 NAME        SIZE                      ATTRIBUTES

$CODE1   000136     47                 RW,I,CON,LCL
$IDATA   000012      5                 RW,D,CON,LCL
$VARS    000006      3                 RW,D,CON,LCL
$TEMPS   000002      1                 RW,D,CON,LCL

TOTAL SPACE ALLOCATED = 000160    56

NO FPP INSTRUCTIONS GENERATED
```

```
        C       BDCONV.FTN
        C       MARCH 13, 1979
        C       BINARY TO DECIMAL CONVERSION ROUTINE
        C       SUBROUTINE BDCONV(INB,LIB,IY)
        C       IY;OUTPUT INTEGER
        C       LIB;LENGTH OF INPUT VECTOR
        C       IY=INB(LIB)+INB(LIB-1)*2+ . . .
0001            SUBROUTINE BDCONV(INB,LIB,IY)
0002            DIMENSION INB(1)
0003            IY=0
0004            IF(LIB.LE.0)RETURN
0005            DO 10 I=1,LIB
0006            IT=2**(I-1)
0007    10      IY=IY+INB(LIB+1-I)*IT
0008            RETURN
0009            END
```

PROGRAM SECTIONS

NAME	SIZE		ATTRIBUTES
$CODE1	000144	50	RW,I,CON,LCL
$IDATA	000012	5	RW,D,CON,LCL
$VARS	000004	2	RW,D,CON,LCL
$TEMPS	000002	1	RW,D,CON,LCL

TOTAL SPACE ALLOCATED = 000164 58

NO FPP INSTRUCTIONS GENERATED

CONV,LP:=CONV/NOTR

```
01234567890123456789     ** RSX-11M V3.2 **     6-OCT-80     16:47:32     DR0:  0
01234567890123456789     ** RSX-11M V3.2 **     6-OCT-80     16:47:32     DR0:[.0
01234567890123456789     ** RSX-11M V3.2 **     6-OCT-80     16:47:32     DR0:[10




PPPPPPPP       AAAAAA      RRRRRRRR       AAAAAA      MM        MM
PPPPPPPP       AAAAAA      RRRRRRRR       AAAAAA      MM        MM
PP      PP   AA      AA    RR      RR   AA      AA    MMMM    MMMM
PP      PP   AA      AA    RR      RR   AA      AA    MMMM    MMMM
PP      PP   AA      AA    RR      RR   AA      AA    MM  MM    MM
PP      PP   AA      AA    RR      RR   AA      AA    MM  MM    MM
PPPPPPPP     AA      AA    RRRRRRRR     AA      AA    MM        MM
PPPPPPPP     AA      AA    RRRRRRRR     AA      AA    MM        MM
PP           AAAAAAAAAA    RR  RR       AAAAAAAAAA    MM        MM
PP           AAAAAAAAAA    RR  RR       AAAAAAAAAA    MM        MM
PP           AA      AA    RR    RR     AA      AA    MM        MM        ....
PP           AA      AA    RR    RR     AA      AA    MM        MM        ....
PP           AA      AA    RR      RR   AA      AA    MM        MM        ....
PP           AA      AA    RR      RR   AA      AA    MM        MM        ....




DDDDDDDD       AAAAAA      TTTTTTTTTT        ;;;;        222222
DDDDDDDD       AAAAAA      TTTTTTTTTT        ;;;;        222222
DD      DD   AA      AA        TT            ;;;;      22      22
DD      DD   AA      AA        TT            ;;;;      22      22
DD      DD   AA      AA        TT                              22
DD      DD   AA      AA        TT                              22
DD      DD   AA      AA        TT            ;;;;            22
DD      DD   AA      AA        TT            ;;;;            22
DD      DD   AAAAAAAAAA        TT            ;;;;          22
DD      DD   AAAAAAAAAA        TT            ;;;;          22
DD      DD   AA      AA        TT              ;;        22
DD      DD   AA      AA        TT              ;;        22
DDDDDDDD     AA      AA        TT            ;;        2222222222
DDDDDDDD     AA      AA        TT            ;;        2222222222




01234567890123456789     ** RSX-11M V3.2 **     6-OCT-80     16:47:32     DR0:[100
01234567890123456789     ** RSX-11M V3.2 **     6-OCT-80     16:47:32     DR0:[  0
01234567890123456789     ** RSX-11M V3.2 **     6-OCT-80     16:47:32     DR0:[100
```

```
+0.69105790E-03          :SBAPC.TBL
-0.14037930E-02          :32 TAP QMF FILTER COEFFICIENT(6400 HZ)
-0.12683030E-02
+0.42341950E-02
+0.14142460E-02
-0.94583180E-02
-0.13038590E-03
+0.17981450E-01
-0.41874830E-02
-0.31238620E-01
+0.14568440E-01
+0.52947450E-01
-0.39348780E-01
-0.99802430E-01
+0.12855790E+00
+0.4664053E+00
+0.4664053E+00
+0.12855790E+00
-0.99802430E-01
-0.39348780E-01
+0.52947450E-01
+0.14568440E-01
-0.31238620E-01
-0.41874830E-02
+0.17981450E-01
-0.13038590E-03
-0.94583180E-02
+0.14142460E-02
+0.42341950E-02
-0.12683030E-02
-0.14037930E-02
+0.69105790E-03
  0.11955E+00    :BETA QUANTIZER WITH 4 BITS
  0.17501E+00
  0.23046E+00
  0.27991E+00
  0.32937E+00
  0.37689E+00
  0.42441E+00
  0.47350E+00
  0.52259E+00
  0.56728E+00
  0.61196E+00
  0.64927E+00
  0.68657E+00
  0.72049E+00
  0.75441E+00
  0.78538E+00
  0.81635E+00
  0.84285E+00
  0.86935E+00
  0.89298E+00
  0.91660E+00
  0.93881E+00
  0.96102E+00
  0.98399E+00
  0.10070E+01
  0.10454E+01
  0.10838E+01
  0.11432E+01
  0.12027E+01
  0.12882E+01
  0.13737E+01
```

```
0.17426E+01    :QQL QUANTIZER WITH 5 BITS
0.18480E+01
0.19535E+01
0.20564E+01
0.21593E+01
0.22642E+01
0.23691E+01
0.24743E+01
0.25795E+01
0.26861E+01
0.27928E+01
0.28983E+01
0.30039E+01
0.31081E+01
0.32122E+01
0.33168E+01
0.34214E+01
0.35233E+01
0.36252E+01
0.37236E+01
0.38220E+01
0.39192E+01
0.40164E+01
0.41137E+01
0.42110E+01
0.43064E+01
0.44018E+01
0.44962E+01
0.45906E+01
0.46848E+01
0.47790E+01
0.48703E+01
0.49616E+01
0.50512E+01
0.51408E+01
0.52307E+01
0.53205E+01
0.54095E+01
0.54985E+01
0.55902E+01
0.56818E+01
0.57713E+01
0.58607E+01
0.59518E+01
0.60428E+01
0.61351E+01
0.62275E+01
0.63214E+01
0.64154E+01
0.65122E+01
0.66090E+01
0.67081E+01
0.68073E+01
0.69119E+01
0.70166E+01
0.71304E+01
0.72442E+01
0.73728E+01
0.75014E+01
0.76496E+01
0.77979E+01
0.80013E+01
0.82047E+01
0.19064E+00    :QQH QUANTIZER WITH 5 BITS
0.32683E+00
0.46302E+00
```

0.58423E+00
0.70543E+00
0.82849E+00
0.95154E+00
0.10760E+01
0.12004E+01
0.13188E+01
0.14372E+01
0.15488E+01
0.16605E+01
0.17730E+01
0.18856E+01
0.19939E+01
0.21022E+01
0.22105E+01
0.23188E+01
0.24193E+01
0.25198E+01
0.26204E+01
0.27210E+01
0.28168E+01
0.29127E+01
0.30101E+01
0.31075E+01
0.32044E+01
0.33013E+01
0.33961E+01
0.34910E+01
0.35889E+01
0.36868E+01
0.37795E+01
0.38722E+01
0.39655E+01
0.40588E+01
0.41555E+01
0.42522E+01
0.43483E+01
0.44444E+01
0.45438E+01
0.46432E+01
0.47414E+01
0.48396E+01
0.49364E+01
0.50332E+01
0.51330E+01
0.52329E+01
0.53378E+01
0.54426E+01
0.55504E+01
0.56582E+01
0.57699E+01
0.58816E+01
0.59983E+01
0.61151E+01
0.62371E+01
0.63591E+01
0.64892E+01
0.66194E+01
0.67673E+01
0.69153E+01
-0.87994E+00 :PCRL(1) QUANTIZER WITH 5 BITS
-0.77363E+00
-0.66732E+00
-0.59522E+00
-0.52312E+00
-0.46606E+00
[illegible]

```
-0.40900E+00
-0.36045E+00
-0.31190E+00
-0.26812E+00
-0.22434E+00
-0.18392E+00
-0.14350E+00
-0.10522E+00
-0.66935E-01
-0.30230E-01
 0.64762E-02
 0.42000E-01
 0.77525E-01
 0.11224E+00
 0.14696E+00
 0.18073E+00
 0.21450E+00
 0.24746E+00
 0.28042E+00
 0.31248E+00
 0.34454E+00
 0.37567E+00
 0.40680E+00
 0.43687E+00
 0.46695E+00
 0.49566E+00
 0.52438E+00
 0.55127E+00
 0.57817E+00
 0.60282E+00
 0.62748E+00
 0.64976E+00
 0.67205E+00
 0.69215E+00
 0.71226E+00
 0.73047E+00
 0.74869E+00
 0.76526E+00
 0.78184E+00
 0.79701E+00
 0.81219E+00
 0.82618E+00
 0.84018E+00
 0.85317E+00
 0.86617E+00
 0.87832E+00
 0.89048E+00
 0.90190E+00
 0.91333E+00
 0.92401E+00
 0.93470E+00
 0.94457E+00
 0.95445E+00
 0.96344E+00
 0.97244E+00
 0.98054E+00
 0.98865E+00
-0.96267E+00     :PCRL(2) QUANTIZER WITH 5 BITS
-0.91995E+00
-0.87723E+00
-0.84365E+00
-0.81007E+00
-0.78072E+00
-0.75137E+00
-0.72398E+00
-0.69659E+00
```

-0.67018E+00
-0.64377E+00
-0.61763E+00
-0.59149E+00
-0.56532E+00
-0.53915E+00
-0.51275E+00
-0.48635E+00
-0.45957E+00
-0.43279E+00
-0.40556E+00
-0.37833E+00
-0.35062E+00
-0.32291E+00
-0.29523E+00
-0.26755E+00
-0.24122E+00
-0.21489E+00
-0.19155E+00
-0.16821E+00
-0.14803E+00
-0.12785E+00
-0.10934E+00
-0.90829E-01
-0.73840E-01
-0.56850E-01
-0.40620E-01
-0.24389E-01
-0.83896E-02
0.76100E-02
0.23901E-01
0.40191E-01
0.57370E-01
0.74550E-01
0.92820E-01
0.11109E+00
0.13205E+00
0.15301E+00
0.17825E+00
0.20349E+00
0.23324E+00
0.26299E+00
0.29826E+00
0.33353E+00
0.37267E+00
0.41181E+00
0.45435E+00
0.49689E+00
0.54353E+00
0.59017E+00
0.64602E+00
0.70187E+00
0.77762E+00
0.85337E+00
-0.72278E+00 :PCRL(3) QUANTIZER WITH 3 BITS
-0.59179E+00
-0.46080E+00
-0.35662E+00
-0.25244E+00
-0.18333E+00
-0.11422E+00
-0.56879E-01
0.46074E-03
0.60810E-01
0.12116E+00
0.20370E+00

0.28624E+00	
0.41861E+00	
0.55098E+00	
-0.75740E+00	:PCRL(4) QUANTIZER WITH 3 BITS
-0.60863E+00	
-0.45986E+00	
-0.36269E+00	
-0.26552E+00	
-0.19364E+00	
-0.12176E+00	
-0.57760E-01	
0.62395E-02	
0.72040E-01	
0.13784E+00	
0.21700E+00	
0.29616E+00	
0.41644E+00	
0.53672E+00	
-0.94739E+00	:PCRH(1) QUANTIZER WITH 4 BITS
-0.89316E+00	
-0.83893E+00	
-0.78660E+00	
-0.73427E+00	
-0.68057E+00	
-0.62687E+00	
-0.56998E+00	
-0.51309E+00	
-0.45476E+00	
-0.39643E+00	
-0.34116E+00	
-0.28589E+00	
-0.23492E+00	
-0.18395E+00	
-0.13749E+00	
-0.91032E-01	
-0.46850E-01	
-0.26668E-02	
0.41811E-01	
0.86288E-01	
0.13477E+00	
0.18325E+00	
0.23942E+00	
0.29559E+00	
0.35880E+00	
0.42201E+00	
0.49611E+00	
0.57021E+00	
0.66087E+00	
0.75153E+00	
-0.94890E+00	:PCRH(2) QUANTIZER WITH 4 BITS
-0.87898E+00	
-0.80906E+00	
-0.75051E+00	
-0.69196E+00	
-0.63959E+00	
-0.58722E+00	
-0.53818E+00	
-0.48914E+00	
-0.44235E+00	
-0.39556E+00	
-0.35059E+00	
-0.30562E+00	
-0.26238E+00	
-0.21914E+00	
-0.17735E+00	
-0.13556E+00	

```
-0.94420E-01
-0.53281E-01
-0.11620E-01
 0.30041E-01
 0.73880E-01
 0.11772E+00
 0.16644E+00
 0.21516E+00
 0.27301E+00
 0.33086E+00
 0.40702E+00
 0.48318E+00
 0.59736E+00
 0.71154E+00
-0.78903E+00    :PCRH(3) QUANTIZER WITH 3 BITS
-0.66820E+00
-0.54737E+00
-0.45189E+00
-0.35641E+00
-0.27235E+00
-0.18829E+00
-0.11008E+00
-0.31868E-01
 0.47160E-01
 0.12619E+00
 0.21560E+00
 0.30501E+00
 0.42609E+00
 0.54717E+00
-0.77081E+00    :PCRH(4) QUNTIZER WITH 3 BITS
-0.64754E+00
-0.52427E+00
-0.43351E+00
-0.34275E+00
-0.26413E+00
-0.18551E+00
-0.11102E+00
-0.36528E-01
 0.40490E-01
 0.11751E+00
 0.20637E+00
 0.29523E+00
 0.41846E+00
 0.54169E+00
 0.57160E+00    :LOW BAND(LB)ERROR SIGNAL QUANTIZER WITH 1-BIT
 0.37732E+00    :LOW BAND ERROR SIGNAL QUANTIZER= 2 BITS
 0.10100E+01
 0.16427E+01
 0.20766E+00    :LB ERROR SIGNAL QUANTIZER=3 BITS
 0.44375E+00
 0.67984E+00
 0.10428E+01
 0.14057E+01
 0.20966E+01
 0.27874E+01
 0.12400E+00    :LB ERROR SIG. QUANTIZER WITH 4 BITS
 0.26440E+00
 0.40480E+00
 0.56670E+00
 0.72870E+00
 0.91980E+00
 0.11110E+01
 0.13444E+01
 0.15778E+01
 0.18776E+01
 0.21773E+01
```

```
0.25971E+01
0.30169E+01
0.37240E+01
0.44311E+01
0.65376E+00   :HIGH BAND(HB) ERROR SIGNAL QUANTIZER WITH 1 BIT
0.38877E+00   :HB ERROR SIGNAL QUANTIZER=2 BITS
0.11427E+01
0.18966E+01
0.20196E+00   :HB ERROR SIGNAL QUANTIZER=3 BITS
0.48833E+00
0.77469E+00
0.11967E+01
0.16188E+01
0.23432E+01
0.30676E+01
0.12400E+00   :LB ERROR SIG. QUANTIZER WITH 4 BITS
0.26440E+00
0.40480E+00
0.56670E+00
0.72870E+00
0.91980E+00
0.11110E+01
0.13444E+01
0.15778E+01
0.18776E+01
0.21773E+01
0.25971E+01
0.30169E+01
0.37240E+01
0.44311E+01
0.25182E+00   : TABLE FOR MCAULEY NOISE SUPPRESSION FACTOR NSF=1
0.27069E+00
0.28459E+00
0.29657E+00
0.30749E+00
0.31776E+00
0.32761E+00
0.33718E+00
0.34659E+00
0.35590E+00
0.36518E+00
0.37447E+00
0.38383E+00
0.39329E+00
0.40289E+00
0.41266E+00
0.42263E+00
0.43285E+00
0.44333E+00
0.45413E+00
0.46527E+00
0.47679E+00
0.48873E+00
0.50113E+00
0.51404E+00
0.52750E+00
0.54157E+00
0.55630E+00
0.57173E+00
0.58794E+00
0.60498E+00
0.62291E+00
0.64179E+00
0.66168E+00
0.68263E+00
0.70468E+00
```

0.72785E+00
0.75212E+00
0.77742E+00
0.80364E+00
0.83053E+00
0.85774E+00
0.88474E+00
0.91079E+00
0.93493E+00
0.95600E+00
0.97285E+00
0.98476E+00
0.99222E+00
0.99749E+00
0.20235E+00 :NSF=2
0.21901E+00
0.23188E+00
0.24336E+00
0.25417E+00
0.26462E+00
0.27490E+00
0.28514E+00
0.29542E+00
0.30581E+00
0.31638E+00
0.32716E+00
0.33822E+00
0.34959E+00
0.36131E+00
0.37342E+00
0.38597E+00
0.39899E+00
0.41253E+00
0.42662E+00
0.44133E+00
0.45667E+00
0.47271E+00
0.48949E+00
0.50704E+00
0.52541E+00
0.54464E+00
0.56476E+00
0.58580E+00
0.60778E+00
0.63068E+00
0.65450E+00
0.67918E+00
0.70465E+00
0.73079E+00
0.75742E+00
0.78433E+00
0.81120E+00
0.83769E+00
0.86334E+00
0.88766E+00
0.91012E+00
0.93019E+00
0.94744E+00
0.96157E+00
0.97258E+00
0.98085E+00
0.98713E+00
0.99244E+00
0.99749E+00
0.14609E+00 :NSF=3
0.15927E+00

0.16990E+00
0.17969E+00
0.18915E+00
0.19852E+00
0.20794E+00
0.21752E+00
0.22733E+00
0.23743E+00
0.24789E+00
0.25874E+00
0.27006E+00
0.28188E+00
0.29426E+00
0.30726E+00
0.32092E+00
0.33531E+00
0.35049E+00
0.36650E+00
0.38342E+00
0.40131E+00
0.42023E+00
0.44023E+00
0.46137E+00
0.48370E+00
0.50726E+00
0.53206E+00
0.55810E+00
0.58537E+00
0.61378E+00
0.64323E+00
0.67356E+00
0.70452E+00
0.73583E+00
0.76710E+00
0.79788E+00
0.82764E+00
0.85584E+00
0.88190E+00
0.90532E+00
0.92567E+00
0.94273E+00
0.95647E+00
0.96718E+00
0.97537E+00
0.98182E+00
0.98730E+00
0.99244E+00
0.99749E+00
0.95694E-01 :NSF=4
0.10510E+00
0.11298E+00
0.12043E+00
0.12780E+00
0.13526E+00
0.14291E+00
0.15084E+00
0.15910E+00
0.16776E+00
0.17687E+00
0.18651E+00
0.19672E+00
0.20757E+00
0.21913E+00
0.23148E+00
0.24468E+00
0.25883E+00

0.27400E+00
0.29031E+00
0.30783E+00
0.32668E+00
0.34697E+00
0.36879E+00
0.39225E+00
0.41743E+00
0.44442E+00
0.47325E+00
0.50395E+00
0.53646E+00
0.57068E+00
0.60640E+00
0.64333E+00
0.68105E+00
0.71902E+00
0.75657E+00
0.79295E+00
0.82736E+00
0.85901E+00
0.88722E+00
0.91148E+00
0.93155E+00
0.94752E+00
0.95980E+00
0.96906E+00
0.97619E+00
0.98204E+00
0.98733E+00
0.99244E+00
0.99749E+00
0.57713E-01 :NSF=5
0.63828E-01
0.69099E-01
0.74206E-01
0.79357E-01
0.84662E-01
0.90195E-01
0.96019E-01
0.10219E+00
0.10876E+00
0.11579E+00
0.12334E+00
0.13148E+00
0.14027E+00
0.14979E+00
0.16014E+00
0.17142E+00
0.18372E+00
0.19718E+00
0.21193E+00
0.22811E+00
0.24589E+00
0.26543E+00
0.28694E+00
0.31058E+00
0.33657E+00
0.36507E+00
0.39624E+00
0.43020E+00
0.46697E+00
0.50649E+00
0.54853E+00
0.59270E+00
0.63838E+00

0.68471E+00
0.73063E+00
0.77491E+00
0.81626E+00
0.85352E+00
0.88574E+00
0.91242E+00
0.93352E+00
0.94951E+00
0.96125E+00
0.96987E+00
0.97650E+00
0.98211E+00
0.98734E+00
0.99244E+00
0.99749E+00
0.32632E-01 :NSF=6
0.36309E-01
0.39556E-01
0.42758E-01
0.46038E-01
0.49466E-01
0.53091E-01
0.56957E-01
0.61109E-01
0.65592E-01
0.70454E-01
0.75748E-01
0.81536E-01
0.87886E-01
0.94876E-01
0.10260E+00
0.11115E+00
0.12065E+00
0.13124E+00
0.14308E+00
0.15634E+00
0.17124E+00
0.18801E+00
0.20692E+00
0.22826E+00
0.25237E+00
0.27958E+00
0.31024E+00
0.34469E+00
0.38318E+00
0.42588E+00
0.47273E+00
0.52342E+00
0.57724E+00
0.63306E+00
0.68929E+00
0.74399E+00
0.79505E+00
0.84055E+00
0.87906E+00
0.90990E+00
0.93327E+00
0.95012E+00
0.96191E+00
0.97026E+00
0.97665E+00
0.98214E+00
0.98734E+00
0.99244E+00
0.99749E+00

0.17592E-01 :NSF=7
0.19676E-01
0.21552E-01
0.23428E-01
0.25374E-01
0.27430E-01
0.29629E-01
0.32000E-01
0.34574E-01
0.37384E-01
0.40465E-01
0.43861E-01
0.47618E-01
0.51792E-01
0.56448E-01
0.61662E-01
0.67525E-01
0.74142E-01
0.81641E-01
0.90174E-01
0.99919E-01
0.11109E+00
0.12395E+00
0.13881E+00
0.15601E+00
0.17600E+00
0.19925E+00
0.22632E+00
0.25782E+00
0.29437E+00
0.33653E+00
0.38473E+00
0.43907E+00
0.49918E+00
0.56392E+00
0.63134E+00
0.69860E+00
0.76237E+00
0.81935E+00
0.86702E+00
0.90422E+00
0.93129E+00
0.94984E+00
0.96212E+00
0.97045E+00
0.97672E+00
0.98216E+00
0.98734E+00
0.99244E+00
0.99749E+00
0.91617E-02 :NSF=8
0.10293E-01
0.11327E-01
0.12373E-01
0.13469E-01
0.14637E-01
0.15898E-01
0.17269E-01
0.18770E-01
0.20424E-01
0.22254E-01
0.24290E-01
0.26566E-01
0.29119E-01
0.31999E-01
0.35261E-01

0.38973E-01
0.43218E-01
0.48096E-01
0.53729E-01
0.60269E-01
0.67902E-01
0.76856E-01
0.87418E-01
0.99940E-01
0.11486E+00
0.13272E+00
0.15418E+00
0.18004E+00
0.21121E+00
0.24873E+00
0.29365E+00
0.34689E+00
0.40892E+00
0.47935E+00
0.55648E+00
0.63696E+00
0.71594E+00
0.78797E+00
0.84844E+00
0.89488E+00
0.92754E+00
0.94879E+00
0.96203E+00
0.97054E+00
0.97677E+00
0.98217E+00
0.98734E+00
0.99244E+00
0.99749E+00
0.46515E-02 :NSF=9
0.52468E-02
0.57977E-02
0.63603E-02
0.69542E-02
0.75925E-02
0.82860E-02
0.90459E-02
0.98841E-02
0.10814E-01
0.11851E-01
0.13014E-01
0.14323E-01
0.15805E-01
0.17491E-01
0.19419E-01
0.21635E-01
0.24196E-01
0.27172E-01
0.30651E-01
0.34744E-01
0.39590E-01
0.45366E-01
0.52299E-01
0.60679E-01
0.70883E-01
0.83397E-01
0.98852E-01
0.11806E+00
0.14205E+00
0.17213E+00
0.20984E+00

0.25690E+00
0.31501E+00
0.38526E+00
0.46738E+00
0.55871E+00
0.65354E+00
0.74380E+00
0.82132E+00
0.88084E+00
0.92164E+00
0.94694E+00
0.96168E+00
0.97054E+00
0.97679E+00
0.98217E+00
0.98734E+00
0.99244E+00
0.99749E+00
0.23164E-02 :NSF=10
0.26222E-02
0.29084E-02
0.32029E-02
0.35160E-02
0.38547E-02
0.42250E-02
0.46332E-02
0.50862E-02
0.55919E-02
0.61595E-02
0.67999E-02
0.75262E-02
0.83540E-02
0.93027E-02
0.10396E-01
0.11663E-01
0.13139E-01
0.14872E-01
0.16917E-01
0.19350E-01
0.22263E-01
0.25781E-01
0.30063E-01
0.35321E-01
0.41836E-01
0.49987E-01
0.60283E-01
0.73414E-01
0.90321E-01
0.11227E+00
0.14095E+00
0.17853E+00
0.22763E+00
0.29104E+00
0.37082E+00
0.46681E+00
0.57453E+00
0.68423E+00
0.78303E+00
0.86041E+00
0.91287E+00
0.94407E+00
0.96105E+00
0.97048E+00
0.97680E+00
0.98217E+00
0.98734E+00

0.99244E+00
0.99749E+00
0.11360E-02 :NSF=11
0.12902E-02
0.14359E-02
0.15870E-02
0.17485E-02
0.19243E-02
0.21175E-02
0.23316E-02
0.25706E-02
0.28388E-02
0.31414E-02
0.34849E-02
0.38767E-02
0.43260E-02
0.48442E-02
0.54453E-02
0.61467E-02
0.69704E-02
0.79443E-02
0.91039E-02
0.10495E-01
0.12178E-01
0.14231E-01
0.16759E-01
0.19904E-01
0.23856E-01
0.28879E-01
0.35340E-01
0.43754E-01
0.54855E-01
0.69687E-01
0.89744E-01
0.11715E+00
0.15481E+00
0.20653E+00
0.27660E+00
0.36845E+00
0.48160E+00
0.60788E+00
0.73059E+00
0.83126E+00
0.90013E+00
0.93985E+00
0.96009E+00
0.97037E+00
0.97680E+00
0.98217E+00
0.98734E+00
0.99244E+00
0.99749E+00
0.55006E-03 :NSF=12
0.62671E-03
0.69974E-03
0.77592E-03
0.85786E-03
0.94745E-03
0.10464E-02
0.11567E-02
0.12803E-02
0.14197E-02
0.15779E-02
0.17583E-02
0.19651E-02
0.22036E-02

0.24803E-02
0.28031E-02
0.31821E-02
0.36300E-02
0.41633E-02
0.48029E-02
0.55762E-02
0.65194E-02
0.76804E-02
0.91239E-02
0.10938E-01
0.13245E-01
0.16216E-01
0.20094E-01
0.25230E-01
0.32138E-01
0.41582E-01
0.54711E-01
0.73261E-01
0.99853E-01
0.13835E+00
0.19412E+00
0.27357E+00
0.38173E+00
0.51642E+00
0.66167E+00
0.79050E+00
0.88178E+00
0.93374E+00
0.95868E+00
0.97018E+00
0.97679E+00
0.98217E+00
0.98734E+00
0.99244E+00
0.99749E+00
0.26349E-03 :NSF=13
0.30110E-03
0.33722E-03
0.37511E-03
0.41608E-03
0.46108E-03
0.51104E-03
0.56693E-03
0.62989E-03
0.70121E-03
0.78249E-03
0.87563E-03
0.98296E-03
0.11073E-02
0.12523E-02
0.14224E-02
0.16233E-02
0.18621E-02
0.21481E-02
0.24934E-02
0.29138E-02
0.34302E-02
0.40710E-02
0.48744E-02
0.58934E-02
0.72020E-02
0.89058E-02
0.11157E-01
0.14180E-01
0.18310E-01

0.24061E-01
0.32233E-01
0.44093E-01
0.61670E-01
0.88231E-01
0.12892E+00
0.19135E+00
0.28486E+00
0.41578E+00
0.57607E+00
0.73501E+00
0.85557E+00
0.92493E+00
0.95665E+00
0.96990E+00
0.97678E+00
0.98217E+00
0.98734E+00
0.99244E+00
0.99749E+00
0.12504E-03 :NSF=14
0.14329E-03
0.16094E-03
0.17957E-03
0.19980E-03
0.22213E-03
0.24702E-03
0.27499E-03
0.30662E-03
0.34261E-03
0.38380E-03
0.43121E-03
0.48609E-03
0.54999E-03
0.62485E-03
0.71311E-03
0.81786E-03
0.94308E-03
0.10939E-02
0.12771E-02
0.15015E-02
0.17791E-02
0.21259E-02
0.25641E-02
0.31244E-02
0.38503E-02
0.48044E-02
0.60784E-02
0.78091E-02
0.10205E-01
0.13593E-01
0.18492E-01
0.25754E-01
0.36806E-01
0.54078E-01
0.81748E-01
0.12686E+00
0.20036E+00
0.31553E+00
0.47749E+00
0.66253E+00
0.81864E+00
0.91222E+00
0.95376E+00
0.96951E+00
0.97675E+00

0.98217E+00
0.98734E+00
0.99244E+00
0.99749E+00
0.58838E-04 :NSF=15
0.67610E-04
0.76155E-04
0.85215E-04
0.95102E-04
0.10606E-03
0.11832E-03
0.13215E-03
0.14786E-03
0.16581E-03
0.18643E-03
0.21027E-03
0.23798E-03
0.27040E-03
0.30854E-03
0.35372E-03
0.40761E-03
0.47237E-03
0.55079E-03
0.64657E-03
0.76461E-03
0.91150E-03
0.10962E-02
0.13313E-02
0.16341E-02
0.20296E-02
0.25538E-02
0.32605E-02
0.42303E-02
0.55886E-02
0.75336E-02
0.10388E-01
0.14695E-01
0.21387E-01
0.32128E-01
0.49948E-01
0.80430E-01
0.13362E+00
0.22567E+00
0.37403E+00
0.57331E+00
0.76772E+00
0.89389E+00
0.94961E+00
0.96895E+00
0.97672E+00
0.98217E+00
0.98734E+00
0.99244E+00
0.99749E+00

APPENDIX F

THE 16 KBPS ADAPTIVE TRANSFORM CODER

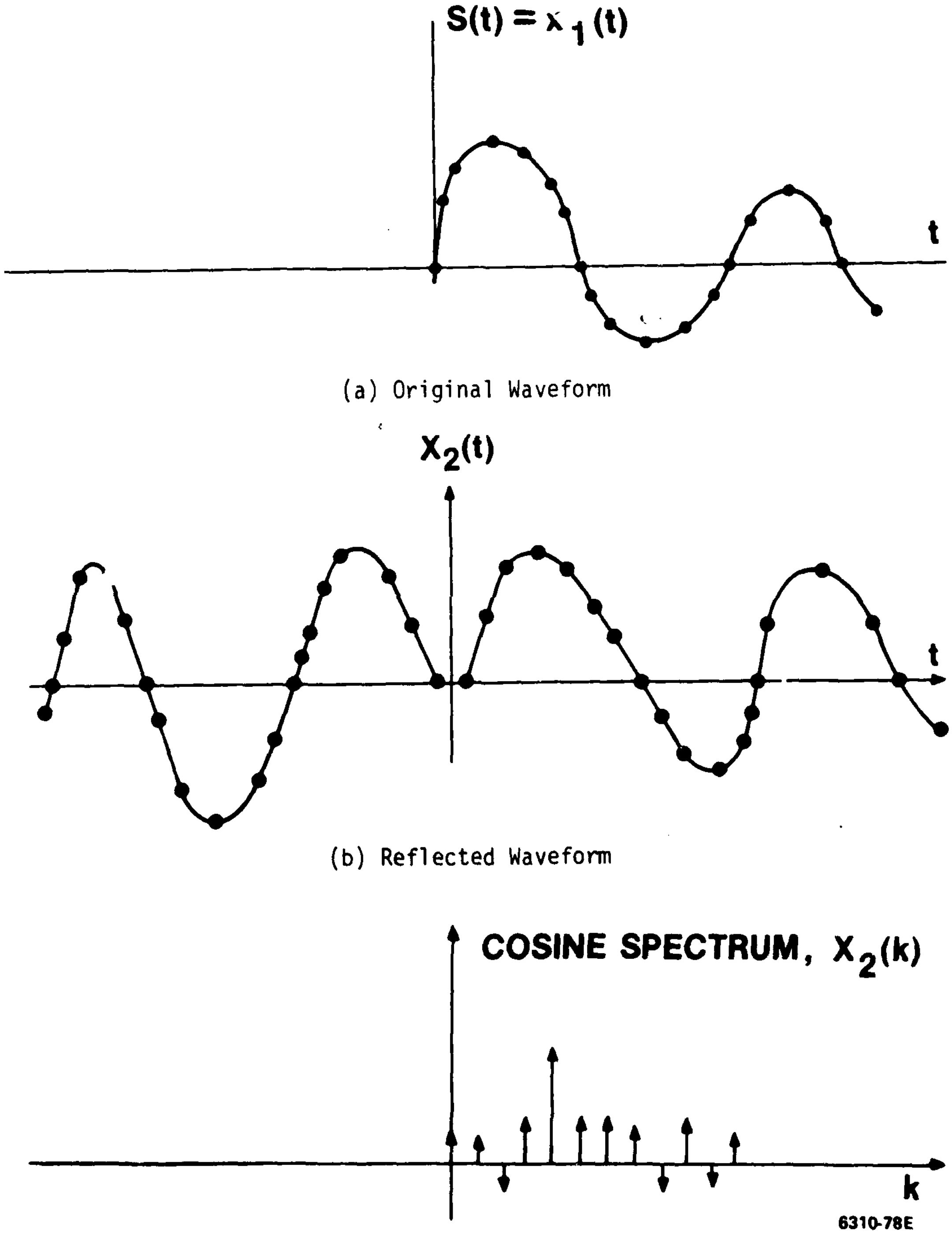
Adaptive Transform Coding (ATC), as proposed by Zelinsky and Noll [1] is an effective block-coding technique for speech encoding in the 8.0 to 16.0 kb/s range.

In its basic form, ATC consists of sending the largest cosine transform coefficients of a segment of data with each coefficient quantized according to an algorithm that gives the larger coefficients more bits than the smaller coefficients. This ATC algorithm departs from earlier algorithms that not only had to send the amplitudes of the coefficients, but also had to send considerable information about which coefficients were quantized and how many bits were associated with each. This extra information could consume as much data capacity as the coefficient amplitudes themselves. Attempts at sending only specific coefficients or the use of a fixed-bit assignment generally reduced voice quality by creating waveform discontinuities at the frame boundaries and by spectrally distorting the signal between boundaries.

In ATC, however, information about which amplitude is sent and how many bits are allocated to each is contained in the basis spectrum, which requires only 2000 to 2400 b/s. This basis spectrum generally is information about the envelope of the transform coefficients being quantized. Its calculation can be performed by the smoothing of transform coefficients or by separate estimates involving least-square analysis [2].

To understand ATC, consider a sampled waveform segment shown in Figure C-1(a). If this waveform is multiplied by 1/2, delayed by half the sampling interval T, and reflected about t=0, it yields $X_2(t)$ whose Fourier transform is given by:

$$X_2(f) = \sum_{n=-(N-1)}^{N-1} x_2(nT)\exp\Big(-j2\pi f(n+1/2)T\Big) \qquad \text{(F-1)}$$



(a) Original Waveform

(b) Reflected Waveform

(c) DFT Output

Figure F-1. Discrete Cosine Transform Operation

If we sample the Fourier transform of $X_2(f)$ at frequencies $\frac{m}{2NT}$, the discrete Fourier transform (DFT) becomes

$$X_2\left(\frac{m}{2NT}\right) = X_2(m) = \sum_{N=-(N-1)}^{N-1} X_2(nT)\ \exp\left(-j\frac{\pi}{N}(n+1/2)\right) \qquad \text{(F-2)}$$

Using symmetry properties of $X_2(nT)$, $X_2(m)$ shown in Figure F-1(b) is real only and is given by

$$X_2(m) = \sum_{n=0}^{N-1} X_1(nT)\ \cos\left(\frac{\pi m}{2N}(2n+1\right) \qquad 0 \le m \le N-1 \qquad \text{(F-3)}$$

Equation (F-3) is the cosine transform. This derivation shows that the Fast Fourier Transform (FFT) can be used to implement the cosine transform by delaying and reflecting the original waveform and then taking the FFT on a waveform twice as long as the original.

The most expensive implementation costs with the ATC algorithm are associated with the Discrete Cosine Transform (DCT) and Discrete Fourier Transform (DFT). Although the DCT cannot be employed directly, methods elaborated by Ahmed et al [3] and Cooley et al [4] use the DFT to compute the desired transform. These algorithms and their interrelationships are shown in Appendix G for clarification. Our FORTRAN simulations are now using the Cooley method for DCT calculation and a special FFT algorithm to lower simulation costs.

After calculation of the ATC coefficients, the basis spectrum (envelope of the cosine transform) can be estimated by making all the cosine transform coefficients positive and smoothing between peaks to efficiently send the envelope. We can quantize the amplitudes of every mth (m is typically 8) envelope sample and send those as the coefficients of the basis spectrum.

However, this original ATC algorithm, as proposed by Zelinski and Noll, suffers from a "burbling" characteristic at lower data rates. To reduce this distortion, Tribolet [2] uses side transmission of pitch and spectral parameters obtained by Linear Predictive Coding (LPC) analysis. The side transmission of the LPC and pitch parameters does in fact remove the "burbling" sound and improve the overall signal-to-noise ratio. Figure F-2 describes the operation of this ATC digitizer.

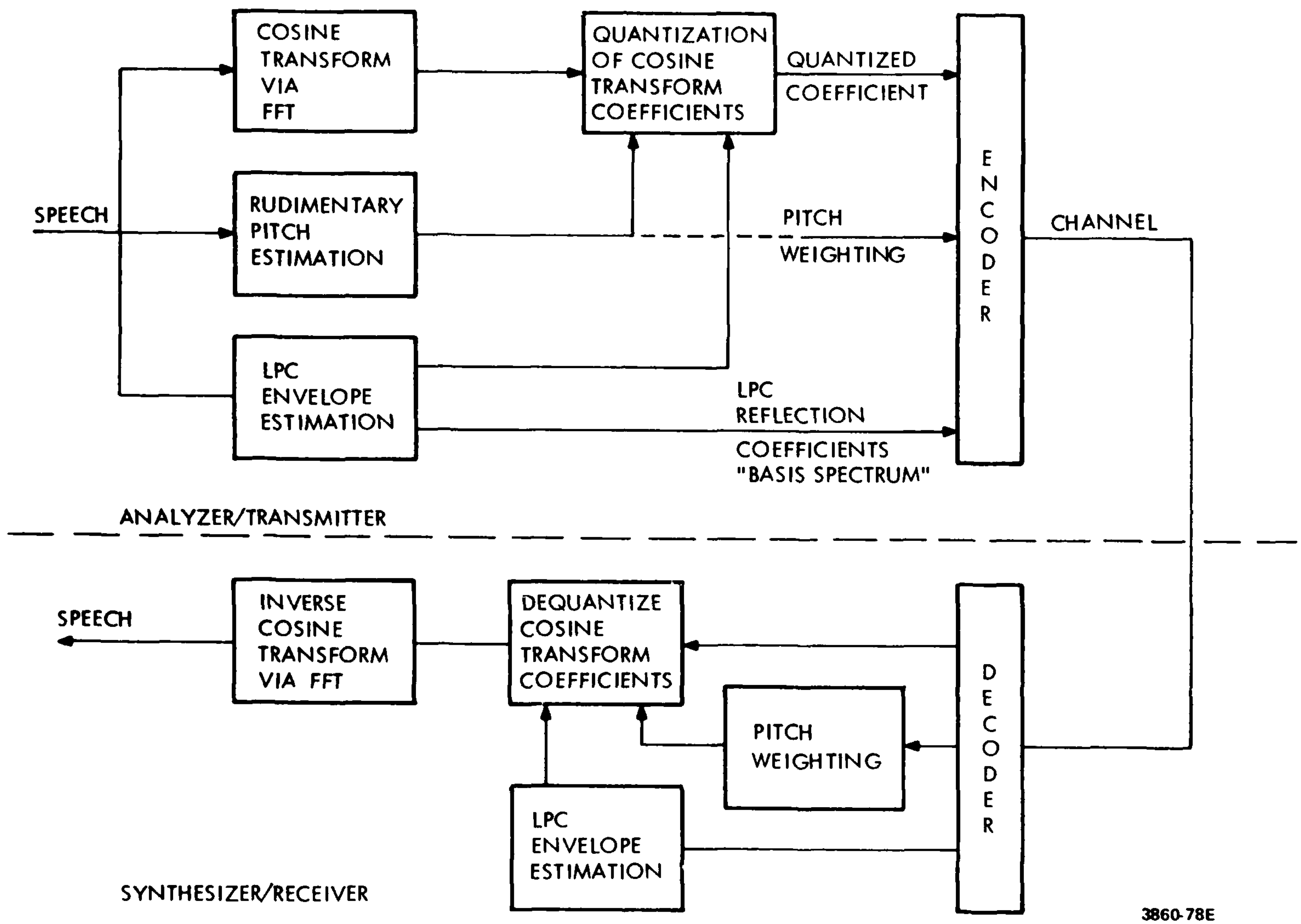


Figure F-2. Adaptive Transform Coder

The innovative solution to the basis spectrum calculation is formed from a least-square analysis of $x_2(t)$, that is, finding those predictor coefficients which minimize.

$$E = \sum_{n=0}^{N-1} \left[x_2(n) - \sum_{i=1}^{P} a_i x_2(n-i) \right]^2 \qquad \text{(F-4)}$$

These predictor coefficients, or alternately reflection coefficients, carry information about the envelope since:

$$y(f) = FFT(a_i) \qquad \text{(F-5)}$$

and the envelope is then $Y^{-1}(f)$.

In addition to linear predictive modeling of the ATC spectrum, the Tribolet approach uses a pitch excitation source. This accounts for the fine structure in the short-time spectrum, which is consistent with the known mechanisms of speech production. This scheme forces the assignment of transform bits to many pitch strictions that otherwise would not be transmitted at all.

With reference to Figure F-3, the ATC analysis is described as follows:

1. The input speech (Figure F-3 (a)) is Fourier transformed to yield a DCT spectrum (Figure F-3 (b)). This spectrum is squared, windowed, and inverse Fourier transformed to yield an autocorrelation function (i.e., pseudo-ACF) of the reflected speech waveform. The first P+1 values of this function are used to define a correlation matrix in the usual normal equation formulation sense. The solution of these equations (i.e., Levinson recursion) yields a prediction filter of order P. The inverse spectrum of this filter yields a smoothed estimate of the DCT (Figure F-3(c)) spectral levels to be used in the adaptation of the quantizers.

2. A rudimentary estimate of the pitch value, M, is found in the pseudo-ACF after the second zero crossing beyond the P+1 ACF value. A corresponding gain factor, G, is also computed as the ratio of ACF(M)/ACF(0). With these two parameters, a pitch pattern is generated in the frequency domain (Figure F-3 (d)) and applied congruently with the LPC spectrum. This combination, yielding a linear prediction spectral fit to the DCT of the input speech, is called the basis spectrum (Figure F-3 (e)).

3. The computation to determine the number of bits to allocate for each transform then proceeds as follows:

Let σ_i be the amplitude of the ith term of the envelope of the basis spectrum. The B_i, the number of bits allocated to the i cosine transform coefficient, is given by:

$$B_i = \left[B_f/N - (1/2N) \sum_{j=1}^{N} \log_2 \sigma_j^2 \right] + 1/2 \log_2 \sigma_i^2 \qquad \text{(F-6)}$$

where

B_f = the total number of bits allocated to send the cosine transform coefficients per frame

N = the total number of cosine transform coefficients calculated per frame.

Note that the term in brackets is calculated once per frame. Fairly simple algorithms ensure that B_i is an integer value and that the sum of the integer B_i adds to B_f.

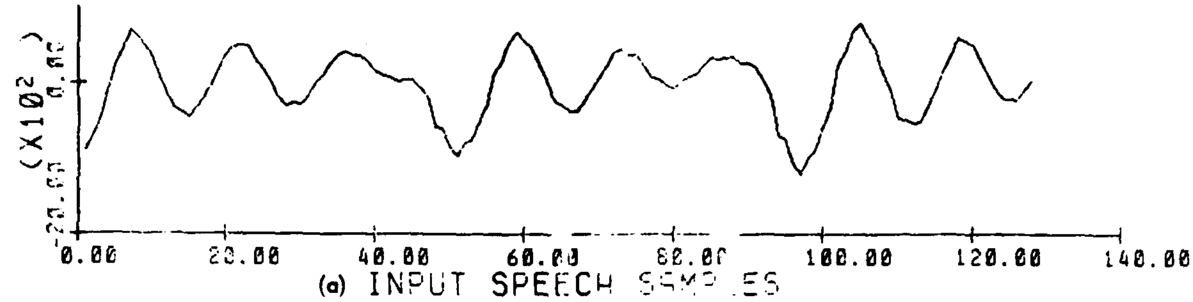
The cosine transform coefficients approximate a Gaussian probability density function. Optimum Gaussian quantizers derived by Max can be used to encode each transform coefficient with B_i bits. Since many of the B_i's will be zero, only larger coefficients are sent. However, GTE has shown that optimal quantizers can be developed that more closely match the transform distribution.

4. The receiver uses the basis spectrum information (LPC, M, G) to regenerate the DCT envelope, to generate the bit allocations using Equation (F-6), to decode the cosine transform coefficients (Figure F-3 (f)), and then to take the inverse cosine transform using the FFT. Frame boundary problems exist at all data rates since quantization of the transform coefficients causes the regenerated waveform to be slightly different than the original. By overlapping the frames slightly and by interpolating across frame boundaries, these discontinuities can be smoothed.

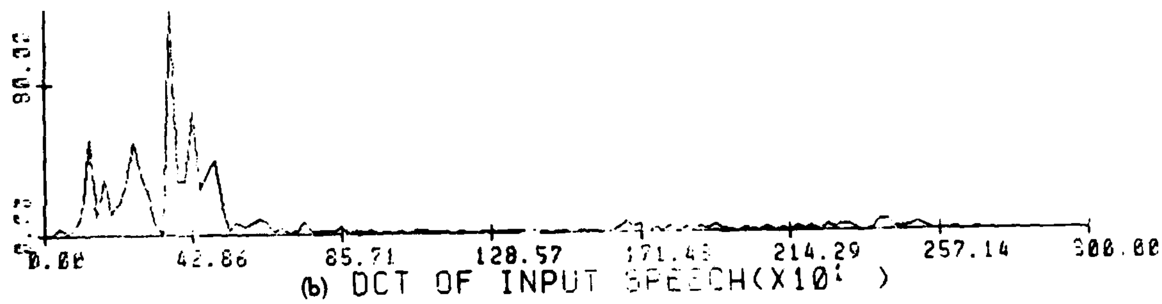
The overall quality of this approach can be surmised from Figure F-3(g), which shows the error waveform defined as:

$$e(n) = s(n) - \hat{s}(n) \qquad \text{(F-7)}$$

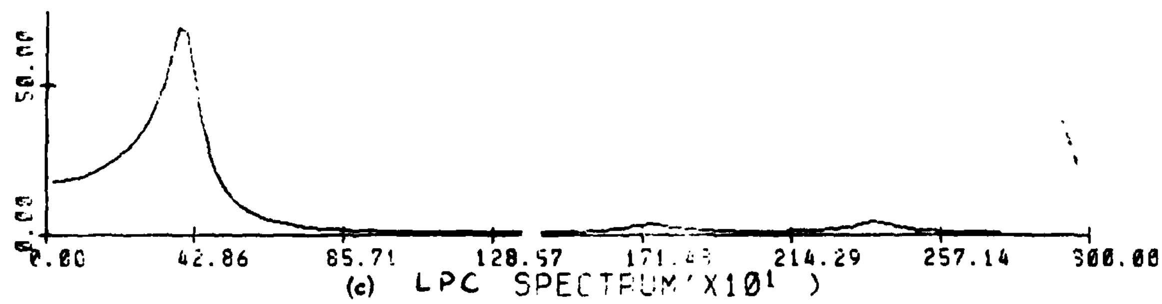
The received waveform, $\hat{s}(n)$, has a high signal-to-noise ratio (~20 dB) for some speakers, even for erroneous pitch estimations made in the analyzer. In fact. GTE has found that an eighth-order LPC predictor (P = 8), coupled with the rudimentary pitch extractor (and voiced/unvoiced logic), yields high quality speech. In summary, the specifications of the 16 Kb/s ATC is shown in Table F-1.



(a) Input Speech Samples



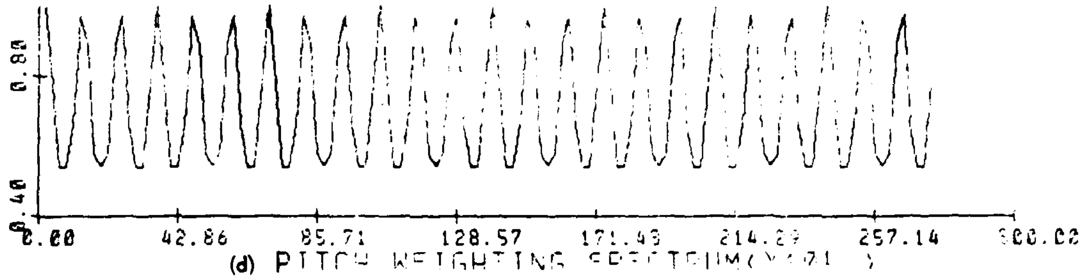
(b) DCT of Input Speech($X10^1$)



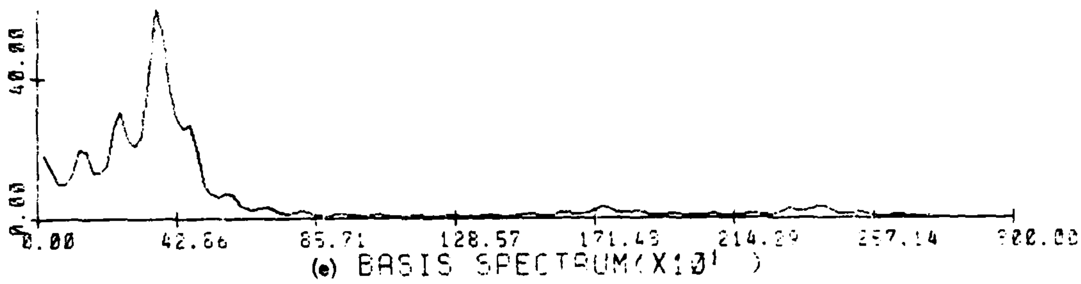
008-79E

(c) LPC Spectrum($X10^1$)

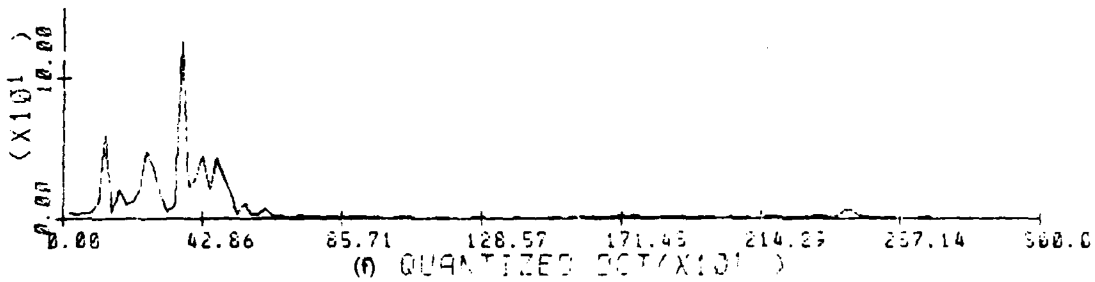
Figure F-3. Graphical Description of Vocoder Strategy for ATC



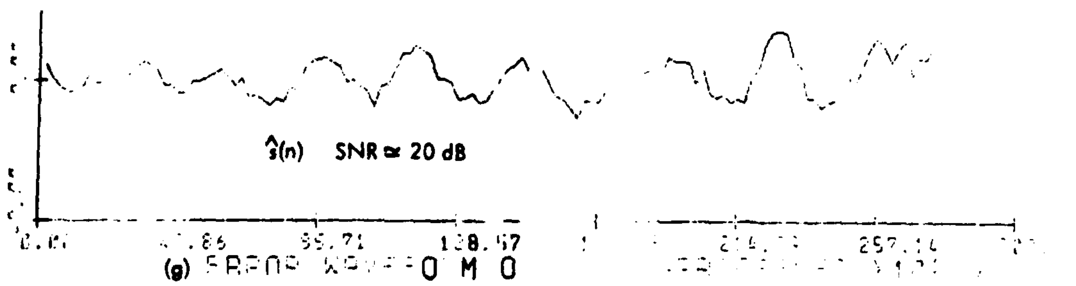
(d) Pitch-Weighting Spectrum(X10^1)



(e) Basic Spectrum(X10^1)



(f) Quantized DCT(X10^1)



(g) Error Waveform-Original-Processed(X10^1)

009-79E

Figure F-3. Graphical Description of Vocoder Strategy for ATC (Cont.)

PARAMETER	SPECIFICATION
Input Bandwidth	0-3200 Hz
Sampling Rate	6400 Hz
Frame Rate	26.016/sec.
Number of Samples/Frame	246
Number of Samples Overlapped/Frame	10
Bits/Frame	615
Pitch	6 if voiced 0 if unvoiced
Pitch Gain	2 if voiced 0 if unvoiced
Voiced/Unvoiced	1
RMS Energy	5
DC BIAS	5
PARCOR 1	5
PARCOR 2	5
PARCOR 3	4
PARCOR 4	4
PARCOR 5	3
PARCOR 6	3
PARCOR 7	2
PARCOR 8	2
Parity Bits (Error Correction)	54
SYNC	1
DCT Coefficients	513 voiced 521 unvoiced
Number of Error Control Blocks/Frame	3
Error Control Technique	(63,45) BCH

TABLE F-1: 16 KBPS ATC SYSTEM SPECIFICATION

REFERENCES for APPENDIX F

[1] R. Zelinski and P. Noll, "Adaptive Transform Coding of Speech Signals," IEEE Trans. Acoustics, Speech and Signal Processing, Vol. ASSP-25, No. 4, August 1977.

[2] J. Tribolet and R. Crochiere, "A Vocoder-Driven Adaptation Strategy for Low-Bit Rate Adaptive Transform Coding of Speech," 1978 International Conference on Digital Signal Processing, Florence, Italy, August 30 to September 2, 1978.

[3] N. Ahmed, T. Nataragan, and K. Rao, "Discrete Cosine Transform," IEEE Trans. Computers, Vol. C-23, 1974, pp. 90-93.

[4] J. Cooley, P. Lewis, and P. Welch, "The Fast Fouries Transform Algorithm: Programming Considerations in the Calculation of Sine, Cosine, and Laplace Transforms," Jour. of Sound Vib., Vol. 12, July 1970, pp. 315-337.

APPENDIX G

INTERRELATIONSHIPS BETWEEN DCT AND DFT ALGORITHMS

Problem Statement: Given a sequence $x(m)$ $m=0,1,\ldots M-1$
formulate the DCT of x as $G_x(k)$ $k=0,1,\ldots M-1$

Solutions:

1. Direct DCT Method

$$G_x(0) = \frac{\sqrt{2}}{M} \sum_{m=0}^{M-1} x(m)$$

$$G_x(k) = \frac{2}{M} \sum_{m=0}^{M-1} x(m) \cos \frac{\pi k(2m+1)}{2m} \quad , \quad k=1,2,\ldots M-1$$

$$\left\{ x(m) \right\} \leftarrow \boxed{\begin{matrix}\text{M-point} \\ \text{DCT}\end{matrix}} \rightarrow \left\{ G_x(k) \right\}$$

2. Ahmed DFT Method

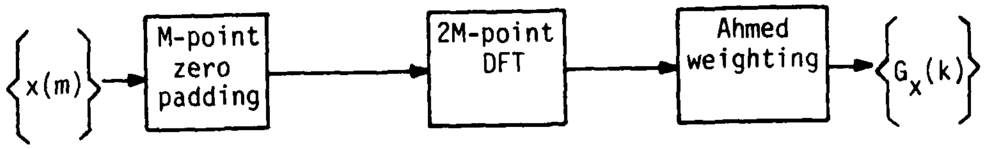
$$G_x(0) = \frac{\sqrt{2}}{M} \overline{X}(0)$$

$$G_x(k) = \frac{2}{M} \operatorname{Re}\left\{ \exp\left(- \frac{j\pi k}{2m} \right) \overline{X}(k) \right\}, \quad k=1,2,\ldots M-1$$

where

$$\overline{X}(k) = \mathrm{DFT}(\hat{x}) = \sum_{m=0}^{2M-1} \hat{x}(m) \exp\left(- \frac{j2\pi km}{2M} \right) , \quad k=0,1,\ldots 2M-1$$

and $\hat{x}(m) = \begin{cases} x(m) & \text{for } m=0,1,\ldots M-1 \\ 0 & \text{for } m=M,M+1,\ldots 2M-1 \end{cases}$



(3) <u>Cooley DFT Method</u>

$$G_x(0) = \frac{\sqrt{2}}{M} \; \underline{\overline{\overline{X}}}(o)$$

$$G_x(k) = \frac{1}{M} \; \mathrm{Re}\left\{ \exp\left(- \frac{j\pi(M-1)k}{2M} \right) \underline{\overline{\overline{X}}}(k) \right\} \; , \; k=1,2,\ldots M-1$$

where $\underline{\overline{\overline{X}}}(k) = \mathrm{DFT}(\hat{x}) = \sum_{m=0}^{2M-1} \hat{x}(m) \; \exp\left(- \frac{j2\pi km}{2M} \right)$

and $\hat{x}(m) = x(M-1-m) + x(m-M)$